Experimentation in Software Engineering
Experimentation in Software Engineering
Foreword

Experimentation is fundamental to any scientific and engineering endeavor. Understanding a discipline involves building models of the various elements of the discipline, e.g., the objects in the domain, the processes used to manipulate those objects, the relationship between the processes and the objects. Evolving domain knowledge implies the evolution of those models by testing them via experiments of various forms. Analyzing the results of the experiment involves learning, the encapsulation of knowledge and the ability to change and refine our models over time. Therefore, our understanding of a discipline evolves over time.

This is the paradigm that has been used in many fields, e.g., physics, medicine, manufacturing. These fields evolved as disciplines when they began applying the cycle of model building, experimenting, and learning. Each field began with recording observations and evolved to manipulating the model variables and studying the effects of changes in those variables. The fields differ in their nature, what constitutes the basic objects of the field, the properties of those objects, the properties of the system that contain them, the relationship of the objects to the system, and the culture of the discipline. These differences affect how the models are built and how experimentation gets done.

Like other science and engineering disciplines, software engineering requires the cycle of model building, experimenting, and learning. The study of software engineering is a laboratory science. The players in the discipline are the researchers and the practitioners. The researcher’s role is to understand the nature of the object (products), the processes that create and manipulate them, and the relationship between the two in the context of the system. The practitioner’s role is to build ‘improved’ systems, using the knowledge available to date. These roles are symbiotic. The researcher needs laboratories to study the problems faced by practitioners and develop and evolve solutions based upon experimentation. The practitioner needs to understand how to build better systems and the researcher can provide models to help.

In developing models and experimenting, both the researcher and the practitioner need to understand the nature of the discipline of software engineering. All software is not the same: there are a large number of variables that cause differences and their
effects need to be understood. Like medicine where the variation in human genetics and medical history is often a major factor in the development of the models and the interpretation of the experiment’s results, software engineering deals with differing contexts that affect the input and the results. In software engineering the technologies are mostly human intensive rather than automated. Like manufacturing the major problem is understanding and improving the relationship between processes and the products they create. But unlike manufacturing, the process in software engineering is development not production. So we cannot collect data from exact repetitions of the same process. We have to build our models at a higher level of abstraction but still take care to identify the context variables.

Currently, there is an insufficient set of models that allow us to reason about the discipline, a lack of recognition of the limits of technologies for certain contexts, and insufficient analysis and experimentation going on but this latter situation is improving as evidenced by this textbook.

This book is a landmark in allowing us to train both the researcher and practitioner in software engineering experimentation. It is a major contribution to the field. The authors have accumulated an incredible collection of knowledge and packaged it in an excellent way, providing a process for scoping, planning, running, analyzing and interpreting, and packaging experiments. They cover all necessary topics from threats to validity to statistical procedures.

It is well written and covers a wide range of information necessary for performing experiments in software engineering. When I began doing experiments, I had to find various sources of information, almost always from other disciplines, and adapt them to my needs as best I could. If I had this book to help me, it would have saved me an enormous amount of time and effort and my experiments would probably have been better.

Professor Victor R. Basili
Foreword

I am honored to be asked to write a foreword for this revision of the authors’ book with the same title that was published in 2000. I have used the original edition since its publication as a teacher and a researcher. Students in my courses at Colorado State University, Washington State University, University of Denver, and Universitaet Wuerzburg have used the book over the years. Some were full-time employees at major companies working on graduate degrees in Systems Engineering, others full-time Masters and Ph.D. students. The book worked well for them. Besides the treatment of experimental software engineering methods, they liked its conciseness. I am delighted to see that the revised version is as compact and easy to work with as the first.

The additions and modifications in this revised version very nicely reflect the maturation of the field of empirical software engineering since the book was originally published: the increased importance of replication and synthesis of experiments, and the need of academics and professionals to successfully transfer new technology based on convincing quantitative evidence. Another important improvement concerns the expanded treatment of ethical issues in software engineering experimentation. Especially since no formal code of ethics exists in this field, it is vitally important that students are made aware of such issues and have access to guidelines how to deal with them.

The original edition of this book emphasized experiments. In industry, however, case studies tend to be more common to evaluate technology, software engineering processes, or artifacts. Hence the addition of a chapter on case studies is much needed and welcomed. So is the chapter on systematic literature reviews.

Having taught a popular quantitative software engineering course with the original edition for a dozen years, this revised version with its additions and updates provides many of the materials I have added separately over the years. Even better, it does so without losing the original edition’s compactness and conciseness. I, for one, am thrilled with this revised version and will continue to use it as a text in my courses and a resource for my student researchers.

Professor Anneliese Amschler Andrews
Foreword from Original Edition

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 Amschler Andrews (formerly von Mayrhauser)
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 valuable tools 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 scientific discipline 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 in 1968, 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 scientific discipline. Before looking at the issues we must address to turn software engineering into science, let us look at the way science is viewed in other areas.
In “Fermat’s Last Theorem” by Dr. Simon Singh, [160], 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 may not be the way most research is conducted in software engineering today. However, the need to evaluate and validate new research proposals by conducting empirical studies is acknowledged to a higher degree today than 10 years ago. 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 empirical 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, although short introductions are provided also for other empirical approaches. The introduction into experimentation 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 few books addressing experimentation in software engineering in particular, and actually no book at all when the original edition of this book was published.

**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 case studies, systematic literature reviews 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. 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 three main parts. The outline of the book is summarized in Table 1, which also shows a mapping to the original edition of this book. The first part provides a general introduction to the area of empirical studies in Chap. 1. It puts empirical studies in general and experiments in particular into a software engineering context. In Chap. 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 and practice. In Chap. 4 we provide an overview of how to conduct systematic literature reviews, to synthesize findings from several empirical studies. Chapter 5 gives an overview of the case studies as a related type of empirical studies. In Chap. 6, the focus is set on experimentation by introducing general experiment process.

Part II has one chapter for each experiment step. Chapter 7 discusses how set the scope for an experiment, and Chap. 8 focuses on the planning phase. Operation of the experiment is discussed in Chaps. 9 and 10 presents some methods for analyzing and interpreting the results. Chapter 11 discusses presentation and packaging of the experiment.

Part III contains two example experiments. In Chap. 12, an example is presented where the main objective is to illustrate the experiment process, and the example in Chap. 13 is used to illustrate how an experiment in software engineering may be reported in a paper.

Some exercises and data are presented in Appendix A. Finally, the book displays some statistical tables in Appendix B. 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.

Exercises

The exercises are divided into four categories, the first presented at the end of each chapter in Parts I and II of the book (Chaps. 1–11), and the other three in Appendix A:

Understanding. Five questions capturing the most important points are provided at the end of each chapter. 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 Chaps. 12–13. 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.
Acknowledgements

This book is based on “Experimentation in Software Engineering: An Introduction”, which was published in year 2000. This new version is both a revision and an extension of the former book. We have revised parts of the book, but also added new material, for example, concerning systematic literature reviews and case study research.

A 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. In particular, we would like to express our sincere gratitude to the readers of “Experimentation in Software Engineering: An Introduction”. Your use of the book has been a great source of inspiration and a motivation to publish the current version. In particular, we would like to thank Mr. Alan Kelon Oliveira de Moraes, Universidade Federal de Pernambuco, Brazil for sending the email that actually triggered the revision of the book. Furthermore, we would like to thank the following individuals for having contributed to the book.

First, we would like to thank the first main external user of the book Prof. 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 Andrews, Denver University, USA and Dr. Khaled El Emam, University of Ottawa, Canada for encouraging us to publish the book in the first place and for valuable comments. We also would like to thank Dr. Lionel Briand, University of Luxembourg, Luxembourg; Dr. Christian Bunse, University of Mannheim, Germany and Dr. John Daly formerly at Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany for providing the data for the example on object-oriented design. Our thanks also to Dr. Thomas Thelin for allowing us to include an experiment he did together with two of the authors of the book.

Early drafts of the book was used and evaluated internally within the Software Engineering Research Group at 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 Dr. 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.

For the current version of the book, we have received valuable input and improvement proposals, and hence we would like to thank the following individuals for their valuable input: Prof. Anneliese Andrews, Denver University, USA; Prof. David Budgen, Durham University, UK; Prof. Barbara Kitchenham, Keele University, UK; Prof. Dieter Rombach and Moinul Islam, University of Kaiserslautern, Germany; Prof. Jürgen Börstler, Dr. Samuel Fricker and Dr. Richard Torkar, Blekinge Institute of Technology, Sweden. Thanks also to Mr. Jesper Runeson for work on the LATEX transformation of 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 in general.

For the chapter on case studies, we are grateful for the feedback to the checklists from the ISERN members and attendants of the International Advanced School of Empirical Software Engineering in September 2007. A special thank to Dr. Kim Weyns and Dr. Andreas Jedlitschka for their review of an early draft of the chapter.

Numerous research projects at Lund University and Blekinge Institute of Technology have over the years contributed to the book. Different grants have funded research projects where empirical studies have been a cornerstone, and hence helped shape our experience that we have tried to document through the book. This book is to some extent a result of all these research projects.

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

Part I  Background

1  Introduction ......................................................... 3
   1.1  Software Engineering Context ............................... 3
   1.2  Science and Software Engineering ......................... 5
   1.3  Exercises ..................................................... 8

2  Empirical Strategies .............................................. 9
   2.1  Overview of Empirical Strategies ........................... 10
   2.2  Surveys .......................................................... 12
      2.2.1  Survey Characteristics ................................ 12
      2.2.2  Survey Purposes ....................................... 13
      2.2.3  Data Collection ......................................... 13
   2.3  Case Studies .................................................. 14
      2.3.1  Case Study Arrangements ............................... 15
      2.3.2  Confounding Factors and Other Aspects ............... 15
   2.4  Experiments ................................................... 16
      2.4.1  Characteristics ......................................... 17
      2.4.2  Experiment Process .................................... 18
   2.5  Empirical Strategies Comparison ............................ 18
   2.6  Replications .................................................. 19
   2.7  Theory in Software Engineering ............................. 21
   2.8  Aggregating Evidence from Empirical Studies .............. 22
   2.9  Empiricism in a Software Engineering Context .......... 24
      2.9.1  Empirical Evaluation of Process Changes .............. 24
      2.9.2  Quality Improvement Paradigm .......................... 26
      2.9.3  Experience Factory ..................................... 27
      2.9.4  Goal/Question/Metric Method .......................... 29
   2.10 Empirically-Based Technology Transfer ..................... 30
   2.11 Ethics in Experimentation ................................... 33
   2.12 Exercises ..................................................... 36
### 10.3 Hypothesis Testing

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.3.1 Basic Concept</td>
<td>132</td>
</tr>
<tr>
<td>10.3.2 Parametric and Non-parametric Tests</td>
<td>135</td>
</tr>
<tr>
<td>10.3.3 Overview of Tests</td>
<td>136</td>
</tr>
<tr>
<td>10.3.4 t-Test</td>
<td>138</td>
</tr>
<tr>
<td>10.3.5 Mann-Whitney</td>
<td>139</td>
</tr>
<tr>
<td>10.3.6 F-Test</td>
<td>140</td>
</tr>
<tr>
<td>10.3.7 Paired t-Test</td>
<td>140</td>
</tr>
<tr>
<td>10.3.8 Wilcoxon</td>
<td>141</td>
</tr>
<tr>
<td>10.3.9 Sign Test</td>
<td>142</td>
</tr>
<tr>
<td>10.3.10 ANOVA (ANalysis Of VAriance)</td>
<td>143</td>
</tr>
<tr>
<td>10.3.11 Kruskal-Wallis</td>
<td>144</td>
</tr>
<tr>
<td>10.3.12 Chi-2</td>
<td>145</td>
</tr>
<tr>
<td>10.3.13 Model Adequacy Checking</td>
<td>148</td>
</tr>
<tr>
<td>10.3.14 Drawing Conclusions</td>
<td>149</td>
</tr>
</tbody>
</table>

### 10.4 Example Analysis

- Exercises: 150

### 10.5 Exercises

- Exercises: 151

### 11 Presentation and Package

- Experiment Report Structure: 153
- Exercises: 157

### Part III Example Experiments

### 12 Experiment Process Illustration

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1 Scoping</td>
<td>161</td>
</tr>
<tr>
<td>12.1.1 Goal Definition</td>
<td>161</td>
</tr>
<tr>
<td>12.1.2 Summary of Scoping</td>
<td>163</td>
</tr>
<tr>
<td>12.2 Planning</td>
<td>163</td>
</tr>
<tr>
<td>12.2.1 Context Selection</td>
<td>163</td>
</tr>
<tr>
<td>12.2.2 Hypothesis Formulation</td>
<td>163</td>
</tr>
<tr>
<td>12.2.3 Variables Selection</td>
<td>165</td>
</tr>
<tr>
<td>12.2.4 Selection of Subjects</td>
<td>165</td>
</tr>
<tr>
<td>12.2.5 Experiment Design</td>
<td>165</td>
</tr>
<tr>
<td>12.2.6 Instrumentation</td>
<td>166</td>
</tr>
<tr>
<td>12.2.7 Validity Evaluation</td>
<td>166</td>
</tr>
<tr>
<td>12.3 Operation</td>
<td>167</td>
</tr>
<tr>
<td>12.3.1 Preparation</td>
<td>167</td>
</tr>
<tr>
<td>12.3.2 Execution</td>
<td>168</td>
</tr>
<tr>
<td>12.3.3 Data Validation</td>
<td>168</td>
</tr>
<tr>
<td>12.4 Analysis and Interpretation</td>
<td>169</td>
</tr>
<tr>
<td>12.4.1 Descriptive Statistics</td>
<td>169</td>
</tr>
<tr>
<td>12.4.2 Data Reduction</td>
<td>172</td>
</tr>
<tr>
<td>12.4.3 Hypothesis Testing</td>
<td>172</td>
</tr>
<tr>
<td>12.5 Summary</td>
<td>173</td>
</tr>
<tr>
<td>12.6 Conclusion</td>
<td>174</td>
</tr>
</tbody>
</table>