

Preface

This book provides guidelines for doing design science in information systems and software engineering research. In design science, we iterate over two activities: designing an artifact that improves something for stakeholders and empirically investigating the performance of an artifact in a context. A key feature of the approach of this book is that our object of study is *an artifact in a context*. The artifacts that we design and study are, for example, methods, techniques, notations, and algorithms used in software and information systems. The context for these artifacts is the design, development, maintenance, and use of software and information systems. Since our artifacts are designed for this context, we should investigate them in this context.

Five major themes run through the book. First, we treat design as well as empirical research as *problem-solving*. The different parts of the book are structured according to two major problem-solving cycles: the design cycle and the empirical cycle. In the first, we design artifacts intended to help stakeholders. In the second, we produce answers to knowledge questions about an artifact in context. This dual nature of design science is elaborated in Part I.

Second, the results of these problem-solving activities are *fallible*. Artifacts may not fully meet the goals of stakeholders, and answers to knowledge questions may have limited validity. To manage this inherent uncertainty of problem-solving by finite human beings, the artifact designs and answers produced by these problem-solving activities must be justified. This leads to great emphasis on the validation of artifact designs in terms of stakeholder goals, problem structures, and artifact requirements in Part II. It also leads to great attention to the validity of inferences in the empirical cycle, treated in Part IV.

Third, before we treat the empirical cycle, we elaborate in Part III on the *structure of design theories* and the role of conceptual frameworks in design and in empirical research. Science does not restrict itself to observing phenomena and reporting about it. That is journalism. In science, we derive knowledge claims about unobserved phenomena, and we justify these fallible claims as well as possible, confronting them with empirical reality and submitting them to the critique of peers.

In this process, we form scientific theories that go beyond what we have observed so far.

Fourth, we make a clear distinction between case-based research and sample-based research. In *case-based research*, we study single cases in sequence, drawing conclusions between case studies. This is a well-known approach in the social sciences. In the design sciences, we take the same approach when we test an artifact, draw conclusions, and apply a new test. The conclusions of case-based research typically are stated in terms of the architecture and components of the artifact and explain observed behavior in terms of mechanisms in the artifact and context. From this, we generalize by analogy to the population of similar artifacts. In *sample-based research*, by contrast, we study samples of population elements and make generalizations about the distribution of variables over the population by means of statistical inference from a sample. Both kinds of research are done in design science. In Part V, we discuss three examples of case-based research methods and one example of a sample-based research method.

Fifth and finally, the appendices of the book contain checklists for the design and empirical research cycles. The checklist for empirical research is generic because it applies to all different kinds of research methods discussed here. Some parts are not applicable to some methods. For example, the checklist for designing an experimental treatment is not applicable to observational case study research. But there is a remarkable uniformity across research methods that makes the checklist for empirical research relevant for all kinds of research discussed here. The method chapters in Part V are all structured according to the checklist.

Figure 1 gives a road map for the book, in which you can recognize elements of the approach sketched above. Part I gives a framework for design science and explains the distinction between design problems and knowledge questions. Design problems are treated by following the design cycle; knowledge questions are answered by following the empirical cycle. As pointed out above, these treatments and answers are fallible, and an important part of the design cycle and empirical cycle is the assessment of the strength of the arguments for the treatments that we have designed and for the answers that we have found.

The design cycle is treated in Part II. It consists of an iteration over problem investigation, treatment design, and treatment validation. Different design problems may require different levels of effort spent on these three activities.

The empirical cycle is treated in Part IV. It starts with a similar triple of tasks as the design cycle, in which the research problem is analyzed and the research setup and inferences are designed and validated. Validation of a research design is in fact checking whether the research setup that you designed will support the inferences that you are planning to make. The empirical cycle continues with research execution, using the research setup, and data analysis, using the inferences designed earlier.

Examples of the entire empirical cycle are given in Part V, where four different research methods are presented:

- In *observational case studies*, individual real-world cases are studied to analyze the mechanisms that produce phenomena in these cases. Cases may be social

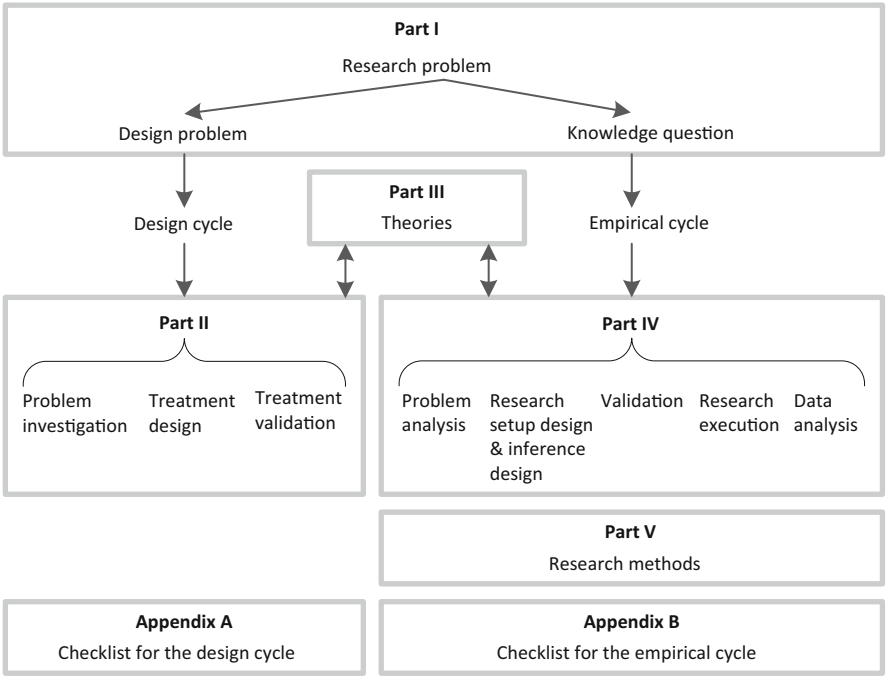


Fig. 1 Road map of the book

- systems such as software projects, teams, or software organizations or they may be technical systems such as complex software systems or networks.
- In *single-case mechanism experiments*, individual cases are experimented with in order to learn which phenomena can be produced by which mechanisms. The cases may be social systems or technical systems, or models of these systems. They are experimented with, and this can be done in the laboratory or in the field. We often speak of *testing* a technical prototype or of *simulating* a sociotechnical system.
 - In *technical action research*, a newly designed artifact is tested in the field by using it to help a client. Technical action research is like single-case mechanism experimentation but with the additional goal of helping a client in the field.
 - In *statistical difference-making experiments*, an artifact is tested by using it to treat a sample of population elements. The outcome is compared with the outcome of treating another sample with another artifact. If there is a statistically discernable difference, the experimenter analyzes the conditions of the experiment to see if it is plausible that this difference is caused, completely or partially, by the difference in treatments.

In the opening chapter of Part V, we return to Fig. 1 and fill in the road map with checklist items. Each research method consists of a particular way of running through the empirical cycle. The same checklist is used for each of them, but not all

items in the checklist are relevant for all methods, and particular items are answered differently for different methods.

The remaining chapters of Part V are about the four research methods and can be read in any order. They give examples of how to use the checklist for different research methods. They are intended to be read when you actually want to apply a research method.

Part III in the middle of the book is about scientific theories, which we will define as generalizations about phenomena that have survived critical assessment and empirical tests by competent peers. Theories enhance our capability to describe, explain, and predict phenomena and to design artifacts that can be used to treat problems. We need theories both during empirical research and during design. Conversely, empirical research as well as design may contribute to our theoretical knowledge.

References to relevant literature are given throughout the book, and most chapters end with endnotes that discuss important background to the chapter. All chapters have a bibliography of literature used in the chapter. The index doubles up as a glossary, as the pages where key terms are defined are printed in boldface.

The book uses numerous examples that have all been taken from master's theses, PhD theses, and research papers.

- Examples are set off from the rest of the text as a bulleted list with square bullets and in a small sans serif typeface.

The first 11 chapters of the book, which cover Parts I–III and the initial chapters of Part IV, are taught every year to master's students of computer science, software engineering, and information systems and an occasional student of management science. A selection of chapters from the entire book is taught every year to PhD students of software engineering, information systems, and artificial intelligence. Fragments have also been taught in various seminars and tutorials given at conferences and companies to academic and industrial researchers. Teaching this material has always been rewarding, and I am grateful for the patience my audiences have had in listening to my sometimes half-baked ideas.

Many of the ideas in the book have been developed in discussions with Hans Heerkens, who knows everything about airplanes as well as about research methods for management scientists. My ideas also developed in work done with Nelly Condori-Fernández, Maya Daneva, Sergio España, Silja Eckartz, Daniel Fernández Méndez, and Smita Ghaisas. The text benefited from comments by Sergio España, Daniel Fernández Méndez, Barbara Paech, Richard Starmans, and Antonio Vetrò.

Last but not least, my gratitude goes to my wife Mieke, who long ago planted the seed for this book by explaining the regulative cycle of the applied sciences to me and who provided a ground for this seed to grow by supporting me when I was endlessly revising this text in my study. My gratefulness cannot be quantified, and it is unqualified.

Enschede, The Netherlands
May 15, 2014

R.J. Wieringa

Design Science Methodology for Information Systems
and Software Engineering

Wieringa, R.J.

2014, XV, 332 p. 43 illus., Hardcover

ISBN: 978-3-662-43838-1