PART I

CASE STUDY METHODOLOGY

CHAPTER 1

INTRODUCTION

1.1 WHAT IS A CASE STUDY?

The term "case study" appears every now and then in the title of software engineering research papers. These papers have in common that they study a specific case, in contrast to a sample from a specified population. However, the presented studies range from very ambitious and well-organized studies in the field of operations (*in vivo*) to small toy examples in a university lab (*in vitro*) that claim to be case studies. This variation creates confusion, which should be addressed by increased knowledge about case study methodology.

Case study is a commonly used research strategy in areas such as psychology, sociology, political science, social work, business, and community planning (e.g., [162, 196, 217]). In these areas, case studies are conducted with the objectives of not only increasing knowledge (e.g., knowledge about individuals, groups, and organizations and about social, political, and related phenomena) but also bringing about change in the phenomenon being studied (e.g. improving education or social care). Software engineering research has similar high-level objectives, that is, to better understand how and why software engineering should be undertaken and, with this knowledge, to seek to improve the software engineering process and the resultant software products.

There are different taxonomies used to classify research in software engineering. The term case study is used in parallel with terms like field study and observational study, each focusing on a particular aspect of the research methodology. For example,

Case Study Research in Software Engineering: Guidelines and Examples, First Edition.
Per Runeson, Martin Höst, Austen Rainer, and Björn Regnell.

Lethbridge et al. use the term *field studies* as the most general term [118], while Easterbrook et al. call *case studies* one of the five "classes of research methods" [47]. Zelkowitz and Wallace propose a terminology that is somewhat different from what is used in other fields, and categorize project monitoring, case study, and field study as *observational methods* [218]. Studies involving change are sometimes denoted *action research* [119, 162, pp. 215–220]. This plethora of terms causes confusion and problems when trying to aggregate multiple empirical studies and to reuse research methodology guidelines from other fields of research.

Yin defines case study as

> an empirical enquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident. [217, p. 13]

This fits particularly well in software engineering. Experimentation in software engineering has clearly shown that there are many factors impacting on the outcome of a software engineering activity, for example, when trying to replicate studies [182]. One of Kitchenham et al.'s [105] preliminary guidelines for empirical research in software engineering states

> Be sure to specify as much of the industrial context as possible. In particular, clearly define the entities, attributes, and measures that are capturing the contextual information.

On the subject of observational studies, which would include case studies, Kitchenham et al. write

> There is an immense variety to be found in development procedures, organizational culture, and products. This breadth implies that empirical studies based on observing or measuring some aspect of software development in a particular company must report a great deal of contextual information if any results and their implications are to be properly understood. Researchers need to identify the particular factors that might affect the generality and utility of the conclusions. [105, p. 723]

Case studies offer an approach that does not require a strict boundary between the object of study and its environment. Case studies do not generate the same results on, for example, causal relationships, as controlled experiments do, but they provide a deeper understanding of the phenomena under study. As they are different from analytical and controlled empirical studies, case studies have been criticized for being of less value, being impossible to generalize from, being biased by researchers, and so on. This critique can be met by applying proper research methodology practices and by reconsidering that knowledge is more than statistical significance [56, 115, 128]. However, the research community has to learn more about the case study methodology in order to conduct, report, review, and judge it properly.

1.2 A BRIEF HISTORY OF CASE STUDIES IN SOFTWARE ENGINEERING

The term *case study* first appeared in software engineering journal papers in the late 1970s. At that time, a case study was typically a *demonstration case*, that is, a case that demonstrated the implementation of some software technology or programming concept.

In the mid- to late-1980s, papers started to report case studies of a broader range of software development phenomena, for example, Alexander and Potter's [3] study of formal specifications and rapid prototying. For these types of papers, the term *case study* refers to a self-experienced and self-reported investigation. Throughout the 1990s the scale of these "self investigations" increased and there were, for example, a series of papers reporting case studies of software process improvement in large and multinational organizations such as Boeing, Hughes, Motorola, NASA, and Siemens.

Case studies based on the external and independent observation of a software engineering activity first appeared in the late 1980s, for example, Boehm and Ross's [23, p. 902] "extensive case study" of the introduction of new information systems into a large industrial corporation in an emerging nation. These case studies, however, did not direct attention at case study methodology that is, at the design, conduct, and reporting of the case study.

The first case study papers that explicitly report the study methodology were published in 1988: Curtis et al.'s [37] field study of software design activities and Swanson and Beath's [199] multiple case study of software maintenance. Given the status of case study research in software engineering at the time, it is not surprising that Swanson and Beath were actually researchers in a school of management in the United States, and were not *software engineering* researchers. Swanson and Beath use their multiple case studies to illustrate a number of challenges that arise when conducting case studies research, and they also present methodological lessons. Their paper therefore appears to be the first of its kind in the software engineering research community that explicitly discusses the challenge of designing, conducting, and reporting case study research.

During the 1990s, both demonstration studies and genuine case studies (as we define them here) were published, although only in small numbers. Glass et al. analyzed software engineering publications in six major software engineering journals for the period 1995–1999 and found that only 2.2% of these publications reported case studies [61]. Much more recently, a sample of papers from Sjøberg et al.'s large systematic review of *experimental* studies in software engineering [195] were analyzed by Holt [72]. She classified 12% of the sample as case studies. This compares to 1.9% of papers classified as formal experiments in the Sjøberg study. But differences in the design of these reviews make it hard to properly compare the reviews and draw firm conclusions.

The first recommendations, by *software engineering researchers*, regarding case study methodology were published in the mid-1990s [109]. However, these recommendations focus primarily on the use of quantitative data. In the late 1990s, Seaman published guidelines on qualitative research [176]. Then, in the early twenty-first

century, a broader set of guidelines on empirical research were published by Kitchenham et al. [105]. Sim et al. arranged a workshop on the topic, which was summarized in *Empirical Software Engineering* [189], Wohlin et al. provided a brief introduction to case studies among other empirical methods [214], and Dittrich et al. edited a special issue of *Information and Software Technology* on qualitative software engineering research [43]. A wide range of aspects of empirical research issues for software engineering are addressed in a book edited by Shull et al. [186]. But the first comprehensive guides to case study research in software engineering were not published until 2009, by Runeson and Höst [170] and Verner et al. [208]. Runeson and Höst's paper was published in the peer-reviewed journal *Empirical Software Engineering* and provides the foundation for this book.

1.3 WHY A BOOK ON CASE STUDIES OF SOFTWARE ENGINEERING?

Case study methodology handbooks are superfluously available in, for example, social sciences [162, 196, 217], which have also been used in software engineering. In the field of information systems (IS) research, the case study methodology is also much more mature than in software engineering. However, IS case studies mostly focus on the information system in its *usage context* and less on the *development and evolution* of information systems. Example sources on case study methodology in IS include Benbasat et al. who provide a brief overview of case study research in information systems [19]. Lee analyzes IS case studies from a positivistic perspective [115] and Klein and Myers do the same from an interpretive perspective [111].

It is relevant to raise the question: what is specific for software engineering that motivates specialized research methodology? In addition to the specifics of the examples, the characteristics of software engineering objects of study are different from social sciences and also to some extent from information systems. The study objects in software engineering have the following properties:

- They are private corporations or units of public agencies *developing* software rather than public agencies or private corporations *using* software systems.
- They are *project*-oriented rather than *line*- or *function*-oriented organizations.
- The studied work is an *advanced engineering work* conducted by highly educated people, rather than a *routine work* [60].
- There is an aim to improve the engineering practices, which implies that there is a component of design research [71] (i.e. prescriptive work).

Sjøberg et al. [194] write that in the typical software engineering situation *actors* apply *technologies* in the performance of *activities* on an existing or planned *software-related* product or interim products. So, for example, requirements analysts (the actors) use requirements engineering tools (the technologies) during requirements elicitation (an activity) to produce a requirements specification (an interim software-related product). Like Pfleeger [139], we use a broad definition of *technology*: any

method, technique, tool, procedure, or paradigm used in software development or maintenance. Sjøberg et al.'s use of the term *actor* is not restricted to mean individual people, but can refer to levels of human behavior. For example, Curtis et al. [37] identified five layers of behavior: the individual, the team, the project, the organization, and the business mileu.

There is a very wide range of activities in software engineering, such as development, operation, and maintenance of software and related artifacts as well as the management of these activities. A frequent aim of software engineering research is to investigate how this development, operation, and maintenance is conducted, and also managed, by software engineers and other stakeholders under different conditions. With such a wide range of activities, and a wide range of software products being developed, there is a very diverse range of skills and experience needed by the actors undertaking these activities.

Software engineering is also distinctive in the *combination* of *diverse* topics that make up the discipline. Glass et al. [60] describe software engineering as an intellectually intensive, nonroutine activity, and Walz et al. [212] describe software engineering as a multiagent cognitive activity. Table 1.1 provides an indication of the topics in the computing field, and therefore the expertise needed by practitioners and researchers.

Many of the interim products are produced either intentionally by the actors (e.g., the minutes of meetings) or automatically by technology (e.g., updates to a version of control system). Therefore, one of the distinctive aspects of software engineering is the raw data that are naturally, and often automatically, generated by the activities and technologies.

There are clear overlaps with other disciplines, such as psychology, management, business, and engineering, but software engineering brings these other disciplines together in a unique way, a way that needs to be studied with research methods tailored to the specifics of the discipline.

Case studies investigate phenomena in their real-world settings, for example, new technologies, communication in global software development, project risk and failure factors, and so on. Hence, the researcher needs to consider not only the practical requirements and constraints from the researcher's perspective, but also the objectives and resource commitments of the stakeholders who are likely to be participating in, or supporting, the case study. Also, practitioners may want to intervene in future projects – that is, change the way things are done in future projects – on the basis of the outcomes from the case studies, and often software engineering managers are interested in *technology* interventions, such as adopting a new technology. This includes both software process improvement (SPI) work [201] and design of solutions [71]. There are, therefore, distinctive practical constraints on case study research in software engineering.

In addition, the software engineering research community has a pragmatic and result-oriented view on research methodology, rather than a philosophical stand, as noticed by Seaman [176]. The community does not pay any larger attention to the inherent conflict between the positivistic foundation for experiments and the interpretive foundation for case studies. This conflict has caused life-long battles in other fields of research. As empirical software engineering has evolved from empirical studies in

TABLE 1.1 Topics in Computing (from Glass et al. [59]).

Problem-solving concepts	Real-time
Algorithms	Edutainment
Mathematics/computational science	
Methodologies	*Systems/software management concepts*
Artificial intelligence	Project/product management
	Process management
Computer Concepts	Measurement/metrics
Computer/hardware principles/	Personnel issues
architecture	Acquisition of software
Intercomputer communication	
Operating systems	*Organizational concepts*
Machine/assembler-level	Organizational structure
data/instructions	Strategy
	Alignment
System/software concepts	Organizational learning/knowledge
System architecture/engineering	management
Software life cycle/engineering	Technology transfer
Programming languages	Change management
Methods/techniques	Information technology implementation
Tools	Information technology usage/operation
Product quality	Management of "computing" function
Human–computer interaction	IT Impact
System security	Computing/information as a business
	Legal/ethical/cultural/
Data/information concepts	
Data/file structures	*Societal concepts*
Database/warehouse/mart organization	Cultural implications
Information retrieval	Legal implications
Data analysis	Ethical implications
Data security	Political implications
Problem domain-specific concepts	*Disciplinary issues*
Scientific/engineering	"Computing" research
Information systems	"Computing" curriculum/teaching
Systems programming	

a natural science context, experimentation and quantitative studies have been considered of higher value compared to case studies and qualitative studies. However, we can observe a slowly growing acceptance for the the case study methodology as a basis for high-quality research, in its contribution to understanding and change in the complex industrial environment of software engineering.

Existing methodology guidelines specifically addressing case studies in software engineering include several publications as presented in Section 1.2 [43, 109, 170,

176, 186, 189, 208, 214]. Still, a comprehensive handbook on *case study research in software engineering* is missing, and that is what this book offers, with *guidelines and examples*.

1.4 CONCLUSION

The term "case study" is used for a broad range of studies in software engineering. There is a need to clarify and unify the understanding of what is meant by a case study, and how a good case study is conducted and reported. There exist several guidelines in other fields of research, but we see a need for guidelines, tailored to the field of software engineering, which we provide in this book.