Chapter 11
Selecting Empirical Methods for Software Engineering Research

Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian

Abstract Selecting a research method for empirical software engineering research is problematic because the benefits and challenges to using each method are not yet well catalogued. Therefore, this chapter describes a number of empirical methods available. It examines the goals of each and analyzes the types of questions each best addresses. Theoretical stances behind the methods, practical considerations in the application of the methods and data collection are also briefly reviewed. Taken together, this information provides a suitable basis for both understanding and selecting from the variety of methods applicable to empirical software engineering.

1. Introduction

Despite widespread interest in empirical software engineering, there is little guidance on which research methods are suitable to which research problems, and how to choose amongst them. Many researchers select inappropriate methods because they do not understand the goals underlying a method or possess little knowledge about alternatives. As a first step in helping researchers select an appropriate method, this chapter discusses key questions to consider in selecting a method, from philosophical considerations about the nature of knowledge to practical considerations in the application of the method. We characterize key empirical methods applicable to empirical software engineering, and explain the strengths and weaknesses of each.

Software engineering is a multi-disciplinary field, crossing many social and technological boundaries. To understand how software engineers construct and maintain complex, evolving software systems, we need to investigate not just the tools and processes they use, but also the social and cognitive processes surrounding them. This requires the study of human activities. We need to understand how individual software engineers develop software, as well as how teams and organizations coordinate their efforts.

Because of the importance of human activities in software development, many of the research methods that are appropriate to software engineering are drawn from disciplines that study human behaviour, both at the individual level (e.g. psychology) and at the team and organizational levels (e.g. sociology).These methods all have known flaws, and each can only provide limited, qualified evidence about the phenomena being studied. However, each method is flawed differently (McGrath, 1995) and viable research strategies use multiple methods, chosen in such a way that the weaknesses of each method are addressed by use of complementary methods (Creswell, 2002).

Describing in detail the wide variety of possible empirical methods and how to apply them is beyond the scope of the chapter. Instead, we identify and compare five classes of research method that we believe are most relevant to software engineering:

- *Controlled Experiments* (including *Quasi-Experiments*)
- *Case Studies* (both *exploratory* and *confirmatory*)
- *Survey Research*
- *Ethnographies*
- *Action Research*

We describe the tradeoffs involved in choosing between these methods, but do not provide a recipe for building research strategies, as we doubt that such recipes exist. The selection of methods for a given research project depends on many local contingencies, including available resources, access to subjects, opportunity to control the variables of interest, and, of course, the skills of the researcher.

To illustrate the steps involved in deciding which method or methods to use, we present two guiding examples. Two fictional software engineering researchers, Joe and Jane, will explore how the various research methods can be applied to their work:

- Jane is a new PhD student interested in the effectiveness of a novel fisheye-view file navigator. Her research is motivated by the fact that navigation is a primary activity of software developers requiring a lot of scrolling and many clicks to find files. "Fisheye-views" use a distortion technique that, if applied correctly, display information in a compact format that could potentially reduce the amount of scrolling required. Jane's intuition is that the fisheye-view file navigator is more efficient for file navigation, but critics argue that the more compact information is difficult to read and that developers will not adopt it over the traditional file navigator. Her research goal, therefore, is to find evidence that supports or refutes her intuition that fisheye-view file navigators are more efficient than traditional file navigators for navigation.
- Joe is a researcher in an industrial lab. His current interests are in understanding how developers in industry use (or not) UML diagrams during software design. This is because, as a student, his professors recommended UML diagrams be used during software design, but his recent exposure to industrial practices indicates that UML is rarely used. His research goal is to explore how widely UML