# 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.



F. Shull et al. (eds.), *Guide to Advanced Empirical Software Engineering*.

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

diagrams are used in industry, and more specifically how these diagrams are used as collaborative shared artefacts during design.

Throughout the remainder of the chapter, we explore how Jane and Joe develop research strategies for their projects. We begin with an analysis of the type of research question(s) they are asking, and the issue of what constitutes valid answers to them. To address the latter question, we tour the main philosophical stances that underpin empirical research. We then describe the five classes of research method, and introduce criteria for distinguishing between them. Along the way, we explore how Jane and Joe might use each method as part of their research strategies. We end the chapter with a look at the practical considerations that affect their choices.

## 2. What kind of Research Question are You Asking?

One of the first steps in choosing an appropriate research method is to clarify the research question. While Jane and Joe have identified the problems they wish to work on, neither has pinned down a precise question. In each case, they could focus on a number of different research questions, each of which leads to a different direction in developing research strategies. The classification of research questions we use in this section is adapted from Meltzoff (1998).

Often, the most obvious question is not the best choice for a starting point. Jane's first attempt to formulate her research question is *"Is a fisheye-view file navigator more efficient than the traditional view for file navigation?"*, while Joe asks *"how widely are UML diagrams used as collaborative shared artifacts during design?"*. Both questions are vague, because they make assumptions about the phenomena to be studied, and kinds of situation in which these phenomena occur. For example, Jane's question only makes sense if we already know that some people (who?) need to do file navigation (whatever that is?), under some circumstances (which are?), and that efficiency (measured how?) is a relevant goal for these people (how would we know that?). Joe's question presupposes that we know what a "collaborative shared artifact" is, and can reliably identify one, and even reliably say which things are UML diagrams. Defining the precise meaning of terms is a crucial part of empirical research, and is closely tied with the idea of developing (or selecting) an appropriate *theory*.

In the early stages of a research program, we usually need to ask *exploratory* questions, as we attempt to understand the phenomena, and identify useful distinctions that clarify our understanding. Suitable research methods for exploratory questions tend to be those that offer rich, qualitative data, which help us to build tentative theories. Unless they are building on existing work that already offers clear definitions, both Jane and Joe need to formulate exploratory questions, such as:

- **Existence questions** of the form, "Does X exist?" Jane might need to ask, *"Is file navigation something that (certain types of programmers) actually do?"* and, *"Is efficiency actually a problem in file navigation?"* Joe might need to ask, *"Do collaborative shared artifacts actually exist?"*
- **Description and Classification questions** such as, "What is X like?", "What are its properties?", "How can it be categorized?", "How can we measure it?", "What is its purpose?", "What are its components?", "How do the components relate to one another?", and "What are all the types of X?" Jane might ask, *"How can we measure efficiency for file navigation?"* and Joe might ask, *"What are all the types of collaborative shared artifacts?"*
- **Descriptive-Comparative questions** of the form, "How does X differ from Y?" investigate similarities and differences between two or more phenomena. Jane might ask, *"How do fisheye views differ from conventional views?"* and Joe might ask, *"How do UML diagrams differ from other representations of design information?"*

The answers to these questions result in a clearer understanding of the phenomena, including more precise definitions of the theoretical terms, evidence that we can measure them, and evidence that the measures are valid. In exploring these questions, Jane and Joe will refine their ideas about the nature of the phenomena they are studying. It is possible that there are already good answers to these questions in the published literature. Jane and Joe must still ask these questions. But a literature survey, instead of an empirical study, may answer them.

Once we have a clearer understanding of the phenomena, we may need to ask *base-rate* questions about the normal patterns of occurrence of the phenomena. If we fail to ask base-rate questions, then we have no basis for saying whether a particular situation is normal or unusual. Example base-rate questions include:

- **Frequency and distribution questions** such as, "How often does X occur?" and, "What is an average amount of X?" Often, these questions can be answered in terms of a standard distribution of a characteristic within a well-defined population. Joe's original question appears to be a frequency question, but there are many ways for him to formulate it more precisely. For example, he might ask, *"How many distinct UML diagrams are created in software development projects in large software companies?"* and he might discover the results follow some standard statistical distribution.
- **Descriptive-Process questions** of the form, "How does X normally work?", "What is the process by which X happens?", "In what sequence do the events of X occur?", "What are the steps X goes through as it evolves?", "How does X achieve its purpose?". For example, Jane might ask, *"How do programmers navigate files using existing tools?"*

Often, we are interested in the *relationship* between two different phenomena, and specifically whether occurrence of one is related to occurrence of the other. Hence we need to formulate some:

- **Relationship questions** such as, "Are X and Y related?" and, "Do occurrences of X correlate with the occurrences of Y?" For example, Jane might ask, *"Does*

*efficiency in file navigation correlate with the programmer's familiarity with the programming environment?"* Joe might ask, *"Do managers' claims about how often they use UML correlate with the actual use of UML?"*

Once we have established that a relationship exists between two phenomena, it is natural to try to explain why the relationship holds by attempting to identify a cause and effect. It is a common mistake to confuse *correlation* with *causality*. In general it is much harder to demonstrate causality than to show that two variables are correlated. If high values of X correlate with high values of Y, it may be because X causes Y, or because Y causes X. But it is also possible that X and Y share some common cause and neither causes the other. Or perhaps they co-evolve in complex ways so that there is no clear cause-and-effect. *Causality* questions include:

- **Causality questions** of the form, "Does X cause Y?" and "Does X prevent Y?" Plus the more general forms: "What causes Y?", "What are all the factors that cause Y?", "What effect does X have on Y?" In software engineering we often ask whether using a particular tool or technique causes an improvement in quality, speed, and so on. Jane's initial question appears to be of this type: *"Do fisheye-views cause an improvement in efficiency for file navigation?"*
- **Causality-Comparative questions** investigate relationships between different causes: "Does X cause more Y than does Z?" or, "Is X better at preventing Y than is Z?" Unless Jane has good base-rate data for existing file navigation tools, Jane's causality question would be better formulated as *"Do fisheye-views cause programmers to be more efficient at file navigation than conventional views?"*
- **Causality-Comparative Interaction** questions investigate how context affects a cause–effect relationship: "Does X or Z cause more Y under one condition but not others?" If Jane's initial studies reveal a factor (e.g., distractions) that affects causality, she might ask *"Do fisheye-views cause programmers to be more efficient at file navigation than conventional views when programmers are distracted, but not otherwise?"*

The classes of research question above are all *knowledge questions* focused on the way the world is. Empirical research in software engineering addresses these types of questions. In contrast, most *non-empirical* research in software engineering focuses on a very different type of question concerned with designing better ways to do software engineering (Simon, 1996):

- **Design questions** of the form, "What's an effective way to achieve X?" or, "What strategies help to achieve X?" For example, Joe's research might lead him to ask, *"What is an effective way for teams to represent design knowledge to improve coordination?"*

These types of question are necessary when the goal is to design better procedures and tools for carrying out some activity or to design suitable social or regulatory policies. Such questions presuppose that the associated knowledge questions have already been addressed so that we have enough information about the nature of the design problem to be solved. In practice, a long term software engineering research

program involves a mix of design questions and knowledge questions as the researchers investigate specific problems, how best to solve them, and which solutions work best (Wieringa and Heerkens, 2006).

## 3. What will You Accept as an Empirical Truth?

Having specified the research question(s), it is worth considering what to accept as *valid* answers. Different people make different assumptions about scientific truth. Take, for example, Jane's causal question: *"Do fisheye-views cause an improvement in efficiency for file navigation?"* Jane's PhD advisor insists that the only trustworthy evidence to answer this question comes from experiments conducted under controlled laboratory conditions, pointing out that the only conclusive way to prove that A causes B is to manipulate A in a controlled setting, and measure the effect on B. However, another member of Jane's thesis committee is an experienced software practitioner and he claims that laboratory experiments are useless, as they ignore the messy complexity of real software projects. He points out that judgments about "improvements" to file navigation are subjective, and contextual factors such as distractions have a major impact. He suggests that Jane should conduct her research *in the field*, investigating what developers actually do on real projects

The different advice Jane receives reflects major differences in opinion over the nature of truth, and how we arrive at it through scientific investigation. The conflicting advice arises from the different *philosophical stances* adopted by members of Jane's committee. To understand the different stances, it helps to know that philosophers make a distinction between *epistemology* (the nature of human knowledge, and how we obtain it) and *ontology* (the nature of the world irrespective of our attempts to understand it). This separation helps us discuss what we accept as scientific knowledge separately from debates about the content of that knowledge (Chalmers, 1999).

Plato originally defined knowledge as *justified true belief*. In other words, to *know* something, you must *believe* it to be true, and have a clear *justification* for believing it to be true. However, epistemologists have argued for centuries about what form that justification should take. Empiricists argue that all knowledge is derived from our experiences and observations of the world, while rationalists argue that some part of our knowledge is innate, hence not derived from experience. Constructivists argue that we cannot separate knowledge from the language we use to express it – because the meanings of words are constructed by social convention, so is our knowledge.

In this chapter we characterize four dominant philosophical stances (Creswell, 2002). The stance you adopt affects which methods you believe lead to acceptable evidence in response to your research question(s). Being explicit about your stance also helps when talking and writing about research. You might not be able to convince other people to change their stance, but you will be able to argue cogently for why you chose the methods you did.

- **Positivitism** states that all knowledge must be based on logical inference from a set of basic observable facts. Positivists are *reductionist*, in that they study things by breaking them into simpler components. This corresponds to their belief that scientific knowledge is built up incrementally from *verifiable* observations, and inferences based on them. Positivism has been much attacked over the past century due to doubts about the reliability of our observations of the world, and the complication that scientific "fact" built up in this manner sometimes turns out to be wrong. While positivism still dominates the natural sciences, most positivists today might more accurately be described as *post-positivists*, in that they tend to accept the idea (due to Popper) that it is more productive to refute theories than to prove them, and we increase our confidence in a theory each time we fail to refute it, without necessarily ever proving it to be true. Positivists prefer methods that start with precise theories from which verifiable hypotheses can be extracted, and tested in isolation. Hence, positivism is most closely associated with the *controlled experiment;* however, *survey research* and *case studies* are also frequently conducted with a positivist stance. Note that a belief in reductionism is needed to accept laboratory experiments as valid in software engineering – you have to convince yourself that the phenomenon you are interested in can be studied in isolation from its context.
- **Constructivism**, also known as *interpretivism* (Klein and Myers, 1999), rejects the idea that scientific knowledge can be separated from its human context. In particular, the meanings of terms used in scientific theories are socially constructed, so interpretations of what a theory means are just as important in judging its truth as the empirical observations on which it is based. Constructivists concentrate less on verifying theories, and more on understanding how different people make sense of the world, and how they assign meaning to actions. Theories may emerge from this process, but they are always tied to the context being studied. For example, an anthropologist studying the culture of a software design team might seek to find out how different members of the team think about and use the tools they have available, and build *local theories* that explain why this particular team uses tools in the way that they do. This stance is often adopted in the social sciences, where positivist/reductionist approaches have little to say about the richness of social interactions. Constructivists prefer methods that collect rich qualitative data about human activities, from which local theories might emerge. Constructivism is most closely associated with *ethnographies*, although constructivists often use e*xploratory case studies* and *survey research* too.
- **Critical Theory** judges scientific knowledge by its ability to free people from restrictive systems of thought (Calhoun, 1995). Critical theorists argue that research is a political act, because knowledge empowers different groups within society, or entrenches existing power structures. Critical theorists therefore choose what research to undertake based on whom it helps. They prefer participatory approaches in which the groups they are trying to help are engaged in the research, including helping to set its goals. Critical theorists therefore tend to take emancipatory or advocacy roles. In sociology, critical theory is most

closely associated with Marxist and feminist studies, along with research that seeks to improve the status of various minority groups. In software engineering, it includes research that actively seeks to challenge existing perceptions about software practice, most notably the open source movement, and, arguably, the process improvement community and the agile community. Critical theorists often use *case studies* to draw attention to things that need changing. However it is a*ction research* that most closely reflects the philosophy of critical theorists.

- **Pragmatism** acknowledges that all knowledge is approximate and incomplete, and its value depends on the methods by which it was obtained (Menand, 1997). For pragmatists, knowledge is judged by how useful it is for solving practical problems. Put simply, truth is whatever works at the time. This stance therefore entails a degree of relativism: what is useful for one person to believe might not be useful for another; therefore truth is relative to the observer. To overcome the obvious criticisms, many pragmatists emphasize the importance of consensus – truth is uncovered in the process of rational discourse, and is judged by the participants as whatever has the better arguments. Pragmatism is less dogmatic than the other three stances described above, as pragmatists tend to think the researcher should be free to use whatever research methods shed light on the research problem. In essence, pragmatism adopts an engineering approach to research – it values practical knowledge over abstract knowledge, and uses whatever methods are appropriate to obtain it. Pragmatists use any available methods, and strongly prefer *mixed methods* research, where several methods are used to shed light on the issue under study.

Although there are examples of research from each of these stances in the software engineering literature, the underlying philosophies are never mentioned. We believe this has contributed to confusion around the selection of empirical methods and appropriate evaluation of empirical research. In particular, it is impossible to avoid some commitment to a particular stance, as you cannot conduct research, and certainly cannot judge its results, without some criteria for judging what constitutes valid knowledge.

## 4. The Role of Theory Building

A distinguishing feature of scientific study is the development of theories that explain how and why certain phenomena occur, and allow predictions to be made. Theories are therefore the building blocks of scientific knowledge. The different philosophical stances differ in their ideas about the role of theory (Gregor, 2006). To the positivist, science is the process of verifying theories by testing hypotheses derived from them. To the constructivist, science is the process of seeking local theories that emerge from (and explain) the data. To the critical theorist, theories are assertions of knowledge (and therefore power), to be critiqued in terms of how

they shape that power. To the pragmatist, theories are the products of a consensual process among a community of researchers, to be judged for their practical utility.

A scientific theory identifies and defines a set of phenomena, and makes assertions about the nature of those phenomena and the relationships between them. A good theory precisely defines the theoretical terms, so that a community of scientists can observe and measure them. A good theory also explains *why* certain relationships occur. Positivists expect their theories to have strong predictive power, and so look for generalized models of cause-and-effect as the basis for theories. In contrast, constructivists expect theories to strengthen their understanding of complex situations, and so tend make more use of categorizations and analogies. Theories are also judged for aesthetic value. Often there is more than one theory that explains empirical observations, so the theories that are simpler, or more elegant are preferred (LittleJohn and Foss, 2004).

As an example, Joe might develop a theory around the use of UML diagrams as a stylized form of external memory. According to his theory, UML diagrams are used to summarize the results of meetings and discussions, to remind participants of a shared understanding that they have already developed. Joe's theory must precisely define the meaning of terms such as "diagram," "participants," "discussions," in order to identify them in any studies performed. Joe's theory should also explain why people choose to use UML in some circumstances but not others, and why they include certain things in their diagrams and exclude others. And finally, it should be able to predict qualities of the diagrams that a software team might produce based on certain factors.

It is important to understand that in any empirical study, theories have a strong impact on how things are observed and interpreted. The theory becomes a "lens" through which the world is observed. This happens whether or not theories are explicitly acknowledged, because real-world phenomena are simply too rich and complex to study without a huge amount of filtering. In *quantitative* research methods, the theoretical lens is used explicitly to decide which variables to isolate and measure, and which to ignore or exclude. In *qualitative* methods, the theoretical lens is often applied after data is collected, to focus the process of labeling and categorizing ("coding") the data.

Few scientists give thought to how theories are created. A notable exception is *Grounded Theory*, a technique for developing theory iteratively from qualitative data (Glaser and Strauss, 1967). In grounded theory, initial analysis of the data begins without any preconceived categories. As interesting patterns emerge, the researcher repeatedly compares these with existing data, and collects more data to support or refute the emerging theory. Despite its close association with the constructivist stance, Grounded Theory probably approximates how most scientists end up developing theories. The difference is that Grounded Theory makes the process explicit and systematic.

Theories also play a role in connecting research to the relevant literature. By defining the key terms, the results of empirical studies can be compared. Furthermore, theories support the process of empirical induction because an individual study can never offer conclusive results. Each study adds more evidence for

or against the propositions of the theory. Without the theory, we have no way of making sense of the accumulation of empirical results.

Software Engineering researchers have traditionally been very poor at making theories explicit (Jørgensen and Sjøberg, 2004). Many of the empirical studies conducted over the past few decades fail to relate the collected data to an underlying theory. The net result is that results are hard to interpret, and studies cannot be compared.

## 5. Selecting Methods

A method is a set of organizing principles around which empirical data is collected and analyzed. A variety of methods can be applied to any research problem, and it is often necessary to use a combination of methods to fully understand the problem. The choice of methods depends upon the theoretical stance of the researcher(s), access to resources (e.g., students or professionals as subjects/participants) and how closely the method aligns with the question(s) that have been posed. Research Design is the process of selecting a method for a particular research problem, tapping into its strengths, while mitigating its weaknesses. The validity of the results depends on how well the research design compensates for the weaknesses of the methods.

Below we describe in more detail the methods most likely to be applied in software engineering contexts. Because these methods are adapted from a number of different fields, there is no consistent terminology to describe them and even a lack of consensus on how to distinguish these methods from one another. We have chosen terms that should be familiar to software engineers and offer definitions and distinctions that capture the spirit of the methods.

### *5.1. Controlled Experiments*

A controlled experiment is an investigation of a testable hypothesis where one or more *independent variables* are manipulated to measure their effect on one or more *dependent variables*. Controlled experiments allow us to determine in precise terms how the variables are related and, specifically, whether a cause–effect relationship exists between them. Each combination of values of the independent variables is a *treatment*. The simplest experiments have just two treatments representing two levels of a single independent variable (e.g. using a tool vs. not using a tool). More complex experimental designs arise when there are more than two levels or more than one independent variable is used. Most software engineering experiments require human *subjects* to perform some task. We measure the effect of the treatments on the subjects.

A precondition for conducting an experiment is a clear hypothesis. The hypothesis (and the theory from which it is drawn) guide all steps of the experimental design,

including deciding which variables to include in the study and how to measure them. For example, Jane might decide to run an experiment to test the hypothesis that fish-eye views *cause* more efficient file navigation than traditional file tree explorer views. This hypothesis is drawn from a theory that explains the effect. The theory is that fisheye views correspond well to the way that people see and navigate in the world, by offering more detail of a specific area of focus, together with a less detailed overview of the peripheral regions, and a smooth way of moving the focus of attention. The theory suggests that less time spent scrolling and fewer clicks should reduce navigation time. This suggests the treatments should be the type of file explorer view used: fisheye view versus the traditional scrolled view, and the dependent variable should be the length of time to navigate to a file.

The theory also helps to decide who the subjects are, and what the tasks should be. To ensure the results of the experiment are valid, the subjects should be drawn from a well-defined population – the idea is to demonstrate that the hypothesis applies to the whole population by testing it on a representative sample. For her experiment, Jane recruits computer science grad students as subject programmers, and screens them to select subjects with lots of programming experience. In SE, it is common to recruit students as subjects. This makes it easier to recruit a large group of subjects, but reduces external validity – an analytical argument is needed for why results on students might still apply to software developers in industry.

Control is important – variables other than the chosen independent variables must not be allowed to affect the experiment. In Jane's case, differences in skill levels of her subjects may affect the experiment, so she might first divide her subjects into groups (or *blocks*) according to their skill level, and randomly assign subjects from each block to the two treatments, for a "between subjects design." An alternative is to use a "within subjects design," in which each subject uses all treatments; however this might introduce learning effects from one treatment to the next, so this needs to be accounted for in the design. Jane needs to decide which confounding factor is more important to control.

The experimental method is closely tied to the positivist stance. This is because experiments are essentially reductionist – they reduce complexity by allowing only a few variables of interest to vary in a controlled manner, while controlling all other variables. If critical variables are ignored or controlled, the experimental results might not generalize to real world settings. For example, in choosing to focus on efficiency as a dependent measure, Jane ignores other possible measures, such as awareness of the file structure that may result from other navigation techniques. The reduction can also mask critical interaction effects, such as the interaction between expertise and preferred navigation environment. For these reasons, if Jane's experiment confirms her hypothesis, it means she has evidence that fish-eye views are more efficient (as she defines efficiency), but it doesn't necessarily mean that fisheye views are better suited to navigation!

The fact that experiments are theory-driven is both a strength and a weakness. It is a strength because basing analysis on hypotheses derived from theories reduces problems of "fishing for results": some correlations occur by chance, and if we look for long enough we'll find them. On the other hand, being theory-driven forces us

to decide in advance which variables to ignore, and they might turn out to be important outside the laboratory setting.

Variants on experiments are possible and can be used in circumstances where a true experiment is not possible. For example, in *quasi-experiments* the subjects are not assigned randomly to the treatments. Quasi-experiments may be used, for example, when, for ethical reasons, subjects must be allowed to choose their treatment. Quasi-experiments are also used in the field. For example if an experiment is performed in a company, there may be constraints on which employees can work on which tasks. In *time-series experiments*, the effect of a treatment is measured in discrete time steps over a period of time. These variations are less powerful than true experiments, and require more careful interpretation.

## *5.2. Case Studies*

There is much confusion in the SE literature over what constitutes a case study. The term is often used to mean a worked example. As an empirical method, a case study is something very different. Yin (2002) introduces the case study as "an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident." Case studies offer in-depth understanding of how and why certain phenomena occur, and can reveal the mechanisms by which cause–effect relationships occur Flyvbjerg (2006). *Exploratory case studies* are used as initial investigations of some phenomena to derive new hypotheses and build theories, and *confirmatory case studies* are used to test existing theories. The latter are especially important for refuting theories: a detailed case study of a real situation in which a theory fails may be more convincing than "failed" experiments in the lab. The detailed insights obtained from confirmatory case studies can also be useful for choosing between rival theories.

A precondition for conducting a case study is a clear research question concerned with how or why certain phenomena occur. This is used to derive a *study proposition* that states precisely what the study is intended to show, and to guide the selection of cases and the types of data to collect. As an example, imagine that Jane is upset as her tool is not adopted by developers after her experiment. She noticed in the post-experiment interviews that subjects frequently mentioned using additional advanced features for navigation that do not involve the file explorer (the only navigation tool available in the experiment). Hence, she poses the research question "How do developers use navigation tool support for large systems under development?", and decides to focus on a specific proposition suggested by the post-experiment interviews that "expert developers use many different strategies for navigation, and move between them very rapidly." This leads her to choose a local company with several very experienced developers as her case, and to focus on observational rather than interview data, to find out what the developers actually do at a fine grain of detail.

The selection of cases is a crucial step in case study research. Case study research uses purposive sampling rather than random sampling. The aim is to select cases that are most relevant to the study proposition. Sometimes a single case is sufficient. This might be because it is a *critical case* for testing a well-formulated theory: if the theory holds for this case, it is likely to be true for many others. Or it might be an *extreme* or *unique case* that is expected to yield interesting insights about what happens under extreme conditions, such as a crisis. Sometimes it is sufficient to identify a *typical case* to gain more insight into common situations. However, a multiple case design usually offers greater validity. The different cases are best thought of as replications, rather than members of a sample. For confirmatory case studies, these can be chosen as *literal replications*, where each case is expected to show the same results, or as *theoretical replications*, where cases are expected to show contrasting results for predictable reasons. An example of the latter would be if Jane's theory predicted that experienced developers do file navigation differently from novices. A multiple case study could include both experts and novices, to confirm that the theory adequately explains both.

A variety of different data sources are typically used in case study research. Qualitative data, including interviews and observation, play a central role, as these offer rich insights into the case. Data collection is always performed with respect to a well-defined *unit of analysis*. In software engineering, the unit of analysis might be a company, a project, a team, an individual developer, a particular episode or event, a specific work product, etc. Choosing an appropriate unit of analysis is important, to ensure the study focuses on the intended phenomena. In Jane's case, she chooses the individual developer as her unit of analysis, allowing her to focus on personal style of different developers. Other choices would lead the case study in different directions. For example, choosing a project as the unit of analysis would allow her to identify whether project teams develop shared navigational styles, but would offer less insights into individual styles. Note that Jane's *case* (a company) has multiple embedded *units of analysis* (the developers). In some studies, the case is the same as the unit of analysis.

Case study research is most appropriate for cases where the reductionism of controlled experiments is inappropriate. This includes situations where the context is expected to play a role in the phenomena (for example if the stresses of a real project affect developers' behaviour), or where effects are expected to be wide ranging, or take a long time (e.g. weeks, months, years) to appear.

The major weakness of case studies is that the data collection and analysis is more open to interpretation and researcher bias. For this reason, an explicit framework is needed for selecting cases and collecting data. Although an individual case study often reveals deep insights, the validity of the results depends on a broader framework of empirical induction. For example, in confirmatory case studies, evidence builds when subsequent case studies also support the theory and/or fail to support rival theories.

Case studies can be applied within all four philosophical stances, although different stances affect the way in which cases are selected and the data analysis is performed. For example, confirmatory case studies draw on the positivist perspective of

theory-driven research, but positivists also use exploratory case studies to develop new theories [see Kitchenham et al. (1995), for an brief tutorial of software engineering case study research using a primarily positivist perspective]. Constructivists use exploratory case studies to investigate the differences of culture and perspective in various settings. Critical theorists use both types of case study to draw attention to situations that are regarded as problematic, selecting cases that are politically important, or for which the participants themselves can be most expected to benefit. The criteria for assessing the validity of a case study depends on which philosophical stance is taken.

## *5.3. Survey Research*

Survey research is used to identify the characteristics of a broad population of individuals. It is most closely associated with the use of questionnaires for data collection. However, survey research can also be conducted by using structured interviews, or data logging techniques. The defining characteristic of survey research is the selection of a *representative sample* from a well-defined *population*, and the data analysis techniques used to *generalize* from that sample to the population, usually to answer base-rate questions.

A precondition for conducting survey research is a clear research question that asks about the nature of a particular target population. Because it is usually infeasible (and unnecessary) to poll every member of that population, survey research first identifies a representative subset as the sample, and determines how to reach that subset for data collection. Identifying the unit of analysis is important for determining an appropriate sampling technique. For example, if the research question is about software companies, then sampling over individual developers may give a biased sample, with some companies being over-represented because several developers from the same company were included. Furthermore, simple random sampling of the population might also be inadequate. For example, if our unit of analysis is individual developers, a random sampling might end up with most or all of respondents working at a single, dominant company. In such a case, stratified sampling techniques would be used, to identify subgroups within the population, so that we can sample within each subgroup.

As an example, recall that Joe wished to understand more about how UML is used in industrial settings, and how UML supports collaborative design. He conducts a survey of software companies across the country to ask them whether they use UML, and if so how. He decides to use individual developers as his unit of analysis, so that he can focus on how different developers perceive the utility of UML. He posts his survey to a number of carefully selected developer email lists, and has a response rate of 10%. The results from the survey are interesting. He discovers that only about 20% of the respondents use UML, and that the diagrams are rarely used in shared settings. He also learns that class diagrams are the most frequently used diagram, with sequence diagrams a close second.

Joe could choose from a number of different designs for his study. For example, if he just wishes to establish how widely UML is used, then he would use a cross-sectional design to obtain a snapshot of participants' current activities. In contrast, a case-control design asks each participant about several related issues in order to establish whether a correlation exists between them, across the population. Joe might use this design if he wishes to explore whether there is a relationship between, say, how long developers have used UML and how much they use it for information sharing. A cohort study tracks changes over time for a group of participants. Joe might use such a design, for example, to determine whether use of UML changes over the life of development project, perhaps with "projects" as his unit of analysis.

A major challenge in survey research is to control for sampling bias. Sampling bias causes problems in generalizing the survey results, because the respondents to the survey may not be representative of the target population. Low response rates increase the risk of bias. For example, if the 10% who responded to Joe's survey were the least busy of his targeted developers, it may be that the survey missed the most skilled, or most senior developers. Or perhaps only people who are frustrated with UML answered his survey. In general, it is hard to obtain high response rates unless significant inducements can be offered for participation, although it is sometimes possible to contact non-respondents to assess whether a systematic response bias has occurred.

An even harder challenge is to ensure that the questions are designed in a way that yields useful and valid data. It can be hard to phrase the questions such that all participants understand them in the same way, especially if the target population is diverse. Also, it is possible that what people say they do in response to survey questions bears no relationship to what they actually do, because they are unable to introspect reliably on their work practices.

It is instructive to compare survey research with other empirical methods. In Joe's case, the survey research design is concerned with establishing what is true of developers in general. If instead he wishes to gain deeper insights into how developers actually use UML, or why they don't, he might be better off conducting a case study. This would sacrifice claims of representativeness (because case studies do not use representative sampling) in return for deeper insights into what happens in a small number of selected cases. On the other hand, if he's more interested in how UML changes how developers share information, he might design an experiment or quasi-experiment to test for a causal relationship.

Survey research falls almost exclusively into the positivist tradition. The desire to characterize an entire population via sampling techniques requires a belief in reductionism, and a concern with generalizable theories. If Joe is more interested in understanding the *culture* of information sharing within development teams, he might instead adopt a constructivist stance, and use ethnography or action research.

Kitchenham and Pfleeger (Chap. 3) provide more detailed information on conducting surveys.

## 5.4. Ethnographies

Ethnography is a form of research focusing on the sociology of meaning through field observation. The goal is to study a *community* of people to understand how the members of that community make sense of their social interactions (Robinson et al., 2007). For software engineering, ethnography can help to understand how technical communities build a culture of practices and communication strategies that enables them to perform technical work collaboratively. An ethnography might focus on a broad technical community (e.g. java programmers in general), or a small, closely knit community (e.g. a single development team).

One notable feature of ethnography is that it avoids imposing any pre-existing theories, but instead focuses on how the members of the community themselves make sense of their social and cultural setting. The researcher explicitly considers his/her own pre-conceptions and how they influence understanding of the studied community. For example, the researcher might focus on phrases used by the community that seem strange to him, to discover how community members use language to create categories that are meaningful to them. The result of an ethnographic study is usually a rich description of the community being studied that helps to build a detailed picture of that community's culture.

The preconditions for an ethnographic study include a research question that focuses on the cultural practices of a particular community, and access to members of that community. Because of the focus on "member's own categories," the precise boundaries of the community to be studied might not be known in advance, and indeed the very notion of membership, and the idea of becoming a member, may be important things to investigate. Using chain sampling, informants within the community are asked to identify representative members of the community, who identify other members of the community, and so on.

As an example, consider the results of the survey that Joe conducted in the previous section. One conclusion from his study is that people don't seem to use UML in the way Joe expected. An ethnography would allow Joe to understand more about how developers use and share UML. He identifies a development team that allows him to observe design meetings for several weeks. He supplements his notes on what he observes with a series of individual and group interviews to further explore how well UML tools match the team's design practices, and why some groups in the company do not use UML.

A special form of ethnography is *participant observation*, where the researcher becomes a member of the community being studied for a period of time. Here, the researcher is not trying to understand the community via the observations of an outsider, but rather through the privileged view that comes from membership. For this to work, the researcher must be accepted by the community as one of them, which may require a much longer duration for the study than "just a few weeks." In software engineering research, becoming a member might only be possible if the researcher has the right technical background.

Ethnographic research takes an explicit constructivist stance. Underlying ethnographic research is the idea that members of a community construct their social and cultural practices on the fly, and their perceptions of those structures also define them. Because of that stance, ethnographic researchers don't seek to prove hypotheses and theories, but rather create *local theories* to improve understanding. This philosophical stance distinguishes ethnography from case studies, surveys and field experiments.

The biggest challenge in ethnographic research is to perform detailed observation, data collection and analysis while avoiding preconceptions. The researcher needs a high degree of training in observational and qualitative data analysis techniques. Sociologists have evolved a collection of techniques for recording observations correctly and for systematic data analysis, as well as for iterative research in which clarifications are sought as new information becomes available. Ethnographic studies in software engineering are valuable for discovering what really goes on in particular (technical) communities, and for revealing subtle but important aspects of work practices.

## *5.5. Action Research*

In Action Research, the researchers attempt to solve a real-world problem while simultaneously studying the experience of solving the problem (Davison et al., 2004). While most empirical research methods attempt to observe the world as it currently exists, action researchers aim to intervene in the studied situations for the explicit purpose of improving the situation. Action research has been pioneered in fields such as education, where major changes in educational strategies cannot be studied without implementing them, and where implementation implies a long term commitment, because the effects may take years to emerge. It has also been adopted in information science, where organizational change can sometimes require a long time to have an impact. However, even in these fields, action research is a relatively new idea, and there is widespread discussion about appropriate methodology, and even debate on the validity of action research as an empirical method.

A precondition for action research is to have a *problem owner* willing to collaborate to both identify a problem, and engage in an effort to solve it. In action research, the problem owners become collaborators in the research. In some cases, the researcher and the problem owner may be the same person. Two key criteria for judging the quality of action research are whether the original problem is *authentic* (i.e. whether it is a real and important problem that needs solving), and whether there are authentic *knowledge outcomes* for the participants. It is additionally important for the researcher to engage in a process of critical reflection upon his past, current and planned actions to identify how they actually helped (or not) to solve the problem. Action research is also characterized by a commitment to effect real change, and an iterative approach to problem solving.

For example, in the process of studying the use of UML, imagine that Joe's colleagues discussed with him their difficulty in integrating software components and predicting the effects of such integration. Joe sees this as an opportunity to work with them to try out ideas from model-driven development (MDD), and to study firsthand how UML changes the way that developers collaborate. Joe initiates a project to work with his colleagues to introduce MDD and to record the experiences. Joe and the development team use a series of data collection techniques, including periodic interviews, questionnaires, and focus groups, to ensure that they establish a process of critical reflection over the life of the project. They use the data collected to develop local theories that explain the experiences of the problem-owners, which, with other research, can be generalized for other people interested in adopting MDD. As new information becomes available, they update these theories to reflect the current understanding of the situation.

Action research is most closely associated with *critical theory*. In an action research project, it is normally taken as self-evident that the problem needs to be solved, and that the adopted solution is desirable: knowledge gained from the research empowers particular individuals or groups, and facilitate a wider change. With this philosophical stance, there is effectively a "moral imperative" to intervene to solve the problem. Therefore, no attempt is made to establish a control group: the moral imperative implies that it would be unethical to withhold the intervention from some groups. Instead, the emphasis is on identifying useful lessons that help others who wish to pursue a similar change agenda. However, action research can be linked to other philosophical stances by divorcing it from its emancipatory roots, and focusing instead on practical problem solving. Positivists would add a concern with careful comparison of the "before" and "after" situations, while constructivists would focus on participants' perceptions of the change process. The key characteristic that differentiates action research from longitudinal case studies and ethnographies is that the researcher is also an agent of change.

The biggest challenge for action research is its immaturity as an empirical method. Although frameworks for evaluating action research have been proposed (e.g. Lau, 1999), they tend to be vague or subjective, leading to accusations that action research is ad hoc. Furthermore, organizational change is often inseparable from organizational politics, and there is a danger that the research fails to address this adequately, either by underestimating the importance of the political agendas of the participants, or by overstating the "moral case" for implementing a change. Researcher bias can be reduced through critical reflection, and by validating the lessons learned through replication. Finally, action research may be expensive, given the organizational commitment needed.

It could be argued that a great deal of software engineering research is actually action research in disguise. Certainly, many key ideas in software engineering were originally developed by trying them out on real development projects, and reporting on the experiences. In this vein, Dittrich (2002) describes cooperative systems development as a form of action research ideally suited to empirical software engineering. By adopting the framework of action research more explicitly, it is likely that the design and evaluation of such research can be made more rigorous. Action research is also

an appealing framework for mixing research with professional activities, especially for practitioners interested in reflecting on their experiences and passing on their learning outcomes for the benefit of others.

## *5.6. Mixed-Methods Approaches*

Throughout this chapter we have seen how Joe and Jane could have used different methods as they learned more about their research topics. While Jane began with the design of an experiment to test the efficiency of file navigation with the fisheye view, she went on to perform a case study to explore some of the unexpected findings from the experiment. This approach can be characterized as *mixed methods* research – a more complex research strategy that emerged in the recognition that all methods have limitations, and the weaknesses of one method can be compensated for by the strengths of other methods (Creswell, 2002).

Mixed method research employs data collection and analysis techniques associated with both quantitative and qualitative data. The "mixing" might be within one study, by using multiple data collection techniques, or among several studies. Key decisions involve the strategy for data collection, and the sequence in which different methods are employed. While mixed method research is a powerful approach to inquiry, the researcher is challenged with the need for extensive data collection, the time-intensive nature of analyzing multiple sources of data, as well as the requirement to be familiar with both quantitative and qualitative forms of research.

We include here the description of three most familiar strategies described by Creswell (2002):

The *Sequential explanatory strategy* is characterized by the collection and analysis of quantitative data followed by the collection and analysis of qualitative data. The purpose of this strategy is typically to use qualitative results to assist in explaining and interpreting the findings of a quantitative study. It is particularly useful when unexpected results arise from the quantitative phase. Jane's example above follows this strategy. When her experimental data indicated that developers switch rapidly between navigation strategies, she decided to perform a case study for a more in-depth exploration of a few developers and their navigation behavior. Damian et al. (2000) provides another example of this approach.

The *Sequential exploratory strategy* is characterized by the collection and analysis of qualitative data followed by the collection and analysis of quantitative data. Its purpose is to use quantitative data and results to assist in the interpretation of qualitative findings. This strategy is also useful for testing elements of an emerging theory resulting from a qualitative study. For example, as a result of Joe's ethnographic study of collaborative design, he formulates some hypotheses about how UML affects the quality of the source code in shared design tasks. To explore this further, he uses a sequential exploratory approach to explore the impact of shared UML diagrams on code quality. He plans and conducts a survey of many different software

development projects, in which he measures the extent to which they use UML for collaboration, and the number of code defects that can be attributed to communication problems. For a published example of this strategy, see Damian and Chisan (2006).

The *Concurrent triangulation strategy* is probably the most familiar and widely used among the mixed-method approaches. This strategy uses different methods concurrently, in an attempt to confirm, cross-validate or corroborate findings. Triangulation is motivated by the fact that often "what people say" could be different than "what people do," and thus collecting data from multiple sources helps improve validity. For example, Joe might incorporate additional data collection techniques into his ethnographic study on the use of UML. He could collect quantitative data from surveys of similar developers to compare against the results of his ethnography. By collecting both types of data simultaneously, rather than sequentially, each analysis can be adapted to explore emerging results from the other. The challenge in this approach is that it may be difficult for the researcher to compare the results of two analyses or to resolve contradictions that arise in the results. In such cases a further source of evidence, or a follow up study might be necessary. For a published example of this strategy, see Bratthall and Jørgensen (2002).

Mixed methods research can be conducted within any of the philosophical stances. For example, a positivist might combine experiments with confirmatory case studies; a constructivist might mix ethnographies with surveys. However, both positivism and constructivism may limit the ability to mix the methods. While positivists strongly prefer quantitative evidence, and constructivists strongly prefer qualitative evidence, mixed methods research emphasizes the use of evidence from both quantitative and qualitative data. Therefore, mixed methods research is more often associated with a pragmatist stance, where the emphasis is on using those methods that most effectively address the research problem.

## 6. Data Collection Techniques

Once the research method has been selected, the researcher must decide which data collection techniques are the most suitable for gathering data based on the study's *unit of analysis*. Multiple techniques can be used to gather data from different perspectives, as there are advantages and limitations to each technique. Indeed, using multiple techniques allows the researcher to *triangulate* even within a single method. If different kinds of data support the same conclusions, it strengthens the study. Singer et al. (Chap. 1) provide an overview of various potential data collection techniques.

Selecting suitable techniques requires careful consideration of the research design as well as the pragmatics of the research setting. It is important to note the advantages and disadvantages of the different techniques from the perspectives of the experimenter, the participants, the generalizability and reliability of the results.

A careful blend of techniques can help to offset potential bias and leads to a more comprehensive understanding of the research topic (Varkevisser et al., 2003). New researchers should ensure they are familiar with the techniques they select, and that they are aware of the potential pitfalls they may face. For example, it is always advisable to pilot-test the data collection instrument, and to pilot-test not just the collection aspect of the instrument, but also the analysis procedure. Many problems do not arise until some data is analyzed and it is often possible to detect such problems with even a small data set. How to analyze the data collected is a topic beyond the scope of this chapter. Wohlin et al. (2000) provide a summary of quantitative analysis techniques for software engineering, and Seaman (Chap. 2) provides an excellent guide to coding etc for qualitative research.

In the end, Jane chose to use a post-study questionnaire that collected both quantitative and qualitative data (open-ended responses). During the study, she observed and videotaped the users and their interactions with the computer so that she could time how long it took to complete the navigation tasks she set for them. She also instrumented the IDE they were using to count number of scrollbar selection events and number of mouse clicks. These numbers can be used with the start/end times indicated on the annotated videotapes of the users. Interviews and focus groups are used at the end of her field study to gather more ideas on how navigation features could be improved in the IDE and why the fisheye view is or is not used by some developers. Joe used questionnaires at different stages in his research. He also conducted interviews and collected observations as a participant in the observed group.

## 7. Empirical Validity

For empirical work to be acceptable as a contribution to scientific knowledge, the researcher needs to convince readers that the conclusions drawn from an empirical study are *valid*. Not surprisingly, the criteria by which researchers judge validity depend on their philosophical stance.

For positivists, research is normally theory-driven. The key steps include deriving study propositions from the theory, designing the study to address the propositions, and then drawing more general conclusions from the results. Each of these steps must be shown to be sound. Accordingly, positivists usually identify four criteria for validity:

- *Construct validity* focuses on whether the theoretical constructs are interpreted and measured correctly. For example, if Jane designs an experiment to test her claims about the efficiency of fish eye views, will she interpret “efficiency” in the same way that other researchers have, and does she have an appropriate means for measuring it? Problems with construct validity occur when the measured variables don’t correspond to the intended meanings of the theoretical terms.

- *Internal validity* focuses on the study design, and particularly whether the results really do follow from the data. Typical mistakes include the failure to handle confounding variables properly, and misuse of statistical analysis.
- *External validity* focuses on whether claims for the generality of the results are justified. Often, this depends on the nature of the sampling used in a study. For example, if Jane's experiment is conducted with students as her subjects, it might be hard to convince people that the results would apply to practitioners in general.
- *Reliability* focuses on whether the study yields the same results if other researchers replicate it. Problems occur if the researcher introduces bias, perhaps because the tool being evaluated is one that the researcher herself has a stake in.

These criteria are useful for evaluating all positivist studies, including controlled experiments, most case studies and survey research. In reporting positivist empirical studies, it is important to include a section on *threats to validity*, in which potential weaknesses in the study design as well as attempts to mitigate these threats are discussed in terms of these four criteria. This is important because all study designs have flaws. By acknowledging them explicitly, the researchers show that they are aware of the flaws and have taken reasonable steps to minimize their effects.

In the constructivist stance, assessing validity is more complex. Many researchers who adopt this stance believe that the whole concept of validity is too positivist, and does not accurately reflect the nature of qualitative research. That is, as the constructivist stance assumes that reality is "multiple and constructed," then repeatability is simply not possible (Sandelowski, 1993). Assessment of validity requires a level of objectivity that is not possible. Attempts to develop frameworks to evaluate the contribution of constructivist research have met with mixed reactions. For example, Lincoln and Guba (1985) proposed to analyze *trustworthiness* of research results in terms of credibility, transferability, dependability, and confirmability. Morse et al. (2002) criticise this as being too concerned with post hoc evaluation, and argue instead for strategies to establish validity during the research process. Creswell (2002) identifies eight strategies for improving validity of constructivist research, which are well suited to ethnographies and exploratory case studies in software engineering:

1. Triangulation: use different sources of data to confirm results and build a coherent picture.
2. Member checking: go back to research participants to ensure that the interpretations of the data make sense from their perspective.
3. Rich, thick descriptions: where possible, use detailed descriptions to convey the setting and findings of the research.
4. Clarify bias: be honest with respect to the biases brought by the researchers to the study, and use this self-reflection when reporting findings.
5. Report discrepant information: when reporting findings, report not only those results which confirm the emerging theory, but also those which appear to present different perspectives on the findings.

6. Prolonged contact with participants: Make sure that exposure to the subject population is long enough to ensure a reasonable understanding of the issues and phenomenon under study.
7. Peer debriefing: Before reporting findings, locate a peer debriefer who can ask questions about the study and the assumptions present in the reporting of it, so that the final account is as valid as possible.
8. External auditor: The same as peer debriefing, except instead of using a person known to the researcher, find an external auditor to review the research procedure and findings.

Dittrich et al. (2007) define a similar set of criteria specifically concerned with validity of qualitative research for empirical software engineering.

For critical theorists, assessment of research quality must also take into account the utility of the knowledge gained. Researchers adopting the critical stance often seek to bring about a change by redressing a perceived injustice, or challenging existing perspectives. Repeatability is not usually relevant, because the problems tackled are context sensitive. The practical outcome is at least as important as the knowledge gained, and any assessment of validity must balance these. However, there is little consensus yet on how best to do this. Lau (1999) offers one of the few attempts to establish some criteria, specifically for action research. His criteria include that the problem tackled should be authentic, the intended change should be appropriate and adequate, the participants should be authentic, and the researchers should have an appropriate level of access to the organization, along with a planned exit point. Most importantly, there should be clear knowledge outcomes for the participants.

## 8. Practical Considerations

In addition to the question of how well the methods fit a given type of research question and philosophical stance, the choice of methods also depends on practical considerations. Often these practical considerations force the researcher to change the original research design in terms of the choice of method, data collection and analysis procedures. It is important to document the original planned research protocol, and all subsequent deviations to it, to allow other researchers to understand the study design, interpret the research results, and replicate the study.

Most of the practical challenges relate to time, budget and personnel resources, and access to data. Rather than describe the challenges for each method individually, we summarize the challenges related to groups of methods, according to the type of data they deal with:

*Methods that are primarily qualitative* include ethnography, case study, and action research. These methods rely on fieldwork, using techniques such as participant observation and interviews. Key challenges include preparing good questions for

structured or semi-structured interviews, and finding the time and resources needed to collect and analyze potentially large sets of data. The researcher needs a thorough training in how to observe and record social behaviour. Access to the field situation may require prolonged time in establishing a relationship with the subject organization such that specific project data is made available. For ethnography, the researcher needs to find a community where she is accepted as a member, which might not be possible unless she has appropriate technical experience. For action research, the researcher needs to balance the need to involve the organization in helping to set appropriate goals for the research with the need to remain objective, such that the research does not become merely consulting.

*Methods that are primarily quantitative* include controlled experiments and survey research. These methods require more significant time in the planning of the research than strictly qualitative methods. To achieve external validity for both experiments and surveys, the researcher needs the time and budget to (1) define, recruit and (if possible) randomly select a sample population that is representative of the target population, (2) design and pilot the questions such that all respondents are presented with questions that they interpret and understand in exactly the same way (therefore careful attention to detail in phrasing the questions is needed), and (3) define statistical tests ahead of time, in order to interpret the collected data. The goal here is to *plan ahead*, for smooth analysis and interpretation of results.

All research conducted in industrial settings brings a number of challenges. It can be very hard to gather data to find out what practitioners actually do, or what needs to be improved in the organization, rather than what practitioners say they do or think require improvement. Data quality can also be an issue (see Chaps. 1 and 7 for more on this issue). In return for access to the organization, the researcher usually has to give up some control. For example, it is hard to observe and document findings without interfering with the observed situation, especially when the industrial partners want to know in advance what the expected outcomes are. It is often difficult to know if changes are made through involvement in the research or would have occurred anyway (c.f., the Hawthorne effect). Finally, obtaining permission to publish the results can be a challenge. Delays in publication are likely if the organization has concerns about inclusion of confidential data or insights in the research. Singer and Vinson (2002) and Vinson and Singer (2004) discuss the unique ethical challenges involved in research in industrial settings.

## 9. Conclusions

We have presented an overview of the choices involved in selecting appropriate empirical methods for software engineering research. Our aim in this chapter was not provide a thorough description of each method, but rather to cover the issues that a researcher must face when deciding how to address a given research problem. Further study, and possibly some specialized training may be necessary before a researcher can apply a chosen method.

We have described the key elements of empirical research design: A clear research question provides a focus to your study. An explicit philosophical stance helps you understand your research goals, and select an appropriate research method. A research method helps you design a study, and decide what kinds of data to collect and how to collect it. A theory helps you explain the data and relate it to the research question and to previous studies in the literature. An appropriate set of criteria for assessing validity helps improve the study design, and clarify the nature of the conclusions.

We have not addressed a number of related topics, including replication and meta-analysis. As the number of empirical studies in software engineering increases, these become more important. In particular, it is only through empirical induction that we come to trust the results of empirical research – i.e. the results need to hold up across many different studies to be considered reliable. Meta-analysis is the process of systematically comparing the results of multiple studies, taking into account differences in the design and context of each individual study. In current software engineering research, meta-analysis is hard to accomplish because of huge variability in the style and quality of the published reports of empirical work.

A key message throughout the chapter is that empirical research never produces certain knowledge. Each of the methods we have available for empirical investigations help to elucidate the phenomena being studied, but each also has significant flaws. Awareness of the limitations of each method should allow you to design a study that minimizes the weaknesses. Furthermore, the flaws can be overcome by mixing methods, and/or by conducting replications (see Brooks et al., Chap. 14, for more information on replication).

We believe that clearer distinctions between research methods are necessary to facilitate better study designs and clearer criteria for evaluating empirical research. The definitions and distinctions we offer in this chapter are by no means widely agreed upon, neither in the empirical software engineering community, nor in related disciplines. For example, we have avoided the usual distinction between qualitative and quantitative methods, as we believe the distinctions between methods are more subtle than simply the type of data collected. Instead, we have emphasized differences in philosophical stance, and in criteria used for designing studies for each type of method. We hope that this chapter provides a first step towards a consensus on empirical methodology in software engineering.

## References

Bratthall, L. and Jørgensen, M. (2002) Can you trust a single data source exploratory software engineering case study? *Journal of Empirical Software Engineering*, 7(1), 9–26.

Calhoun, C. (1995) *Critical Social Theory: Culture, History, and the Challenge of Difference*. Blackwell, Oxford, UK.

Chalmers, A. (1999) *What Is This Thing Called Science?* 3rd Edition, Hackett Publishing Co, Indianapolis.

Creswell, J.W. (2002) *Research Design: Qualitative, Quantitative and Mixed Methods Approaches*. 2nd Edition, Sage Publications, Thousand Oaks, CA.

Damian, D. and Chisan, J. (2006) An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality and risk management, *IEEE Transactions on Software Engineering*, 32(8), 433–453.

Damian, D.E., Eberlein, A., Shaw, M., and Gaines, B. (2000) Using different communication media in requirements negotiation, *IEEE Software*, 17(3), 28–36.

Davison, R.M., Martinsons, M.G., and Kock, N. (2004) Principles of canonical action research, *Information Systems Journal*, 14(1), 65–86.

Dittrich, Y. (2002) Doing Empirical Research on Software Development: Finding a Path Between Understanding, Intervention, and Method Development. In *Social Thinking: Software Practice*, Y. Dittrich, C. Floyd, and R. Klischewski, Eds. MIT Press.

Dittrich, Y., John, M., Singer, J., and Tessem, B. (2007) Editorial for the Special Issue on Qualitative Software Engineering Research, *Information and Software Technology*, 49(6), 531–539.

Flyvbjerg, B. (2006) Five misunderstandings about case study research, *Qualitative Inquiry*, 12(2), 219–245.

Glaser, B.G. and Strauss, A. (1967) *Discovery of Grounded Theory: Strategies for Qualitative Research*. Sociology Press, Mill Valley, CA.

Gregor, S. (2006) The Nature of Theories in Information Systems, *MIS Quarterly*, 30(3), 611–642

Jørgensen, M. and Sjøberg, D.I.K. (2004) Generalization and Theory-Building in Software Engineering Research. *IEE Proceedings, Workshop on Empirical Assessment in Software Engineering* (EASE'04), at ICSE'04, pp. 29–36.

Kitchenham, B., Pickard, L., and Pfleeger, S.L. (1995) Case studies for method and tool evaluation, *IEEE Software*, 12(4), 52–62.

Klein, H.K. and Myers, M.D. (1999) A set of principles for conducting and evaluating interpretive field studies in information systems, *MIS Quarterly*, 23(1), 67–93.

Lau, F. (1999).Towards a framework for action research in information systems studies, *Information Technology and People*, 12(2), 148–175.

Lincoln, Y.S. and Guba, E.G. (1985) *Naturalistic Inquiry*. Sage, Beverly Hills, CA.

Littlejohn, S.W. and Foss, K.A. (2004) *Theories of Human Communication*. 8th Edition, Wadsworth Publishing, Belmont, CA.

McGrath, J.E. (1995) Methodology matters: doing research in the behavioral and social sciences. In *Human–Computer Interaction: Toward the Year 2000*, R.M. Baecker, J. Grudin, W. Buxton, A., and Greenberg, S., Eds. Morgan Kaufmann Publishers, San Francisco, CA, pp. 152–169.

Meltzoff, J. (1998) *Critical Thinking About Research: Psychology and Related Fields*. American Psychological Association, Washington DC.

Menand, L. (1997) *Pragmatism: A Reader*. Vintage Press, New York.

Morse, J.M., Barrett, M., Mayan, M., Olson, K. and Spiers, J. (2002) Verification strategies for establishing reliability and validity in qualitative research, *International Journal of Qualitative Methods*, 1(2), 1–19.

Robinson, H., Segal, J. and Sharp, H. (2007) Ethnographically-informed empirical studies of software practice, *Information and Software Technology*, 49(6), 540–551.

Sandelowski, M. (1993) Rigor or rigor mortis: the problem of rigor in qualitative research revisited, *Advances in Nursing Science*, 16(2), 1–8.

Simon, H. (1996) *The Sciences of the Artificial*. 3rd Edition, MIT Press, Cambridge, MA.

Singer, J.A. and Vinson, N.G. (2002) Ethical issues in empirical studies of software engineering, *IEEE Transactions on Software Engineering*, 28(12), 1171–1180.

Varkevisser, C.M., Pathmanathan, I., and Brownlee, A. (2003) *Designing and Conducting Health Systems Research Projects: Volume 1 – Proposal Development and Fieldwork*. Chapter 10: Data Collection Techniques. Available online at http://www.idrc.ca/en/ev-56605-201-1-DO_TOPIC.html

Vinson, N.G. and Singer, J.A. (2004) Consent issues raised by observational research in organisations, *NCEHR Communique*, 12(2), 35–36.

Wieringa, R.J. and Heerkens, J.M.G. (2006) The methodological soundness of requirements engineering papers: a conceptual framework and two case studies, *Requirements Engineering Journal*, 11, 295–307.

Wohlin, C., Runesson, P., Höst, M., Ohlsson, M.C., Regnell, B., and Wesslén, A. (2000) *Introduction to Experimentation in Software Engineering*. Kluwer Academic Publishers, Boston, MA.

Yin, R.K. (2002) *Case Study Research: Design and Methods*. Sage, Thousand Oaks, CA.