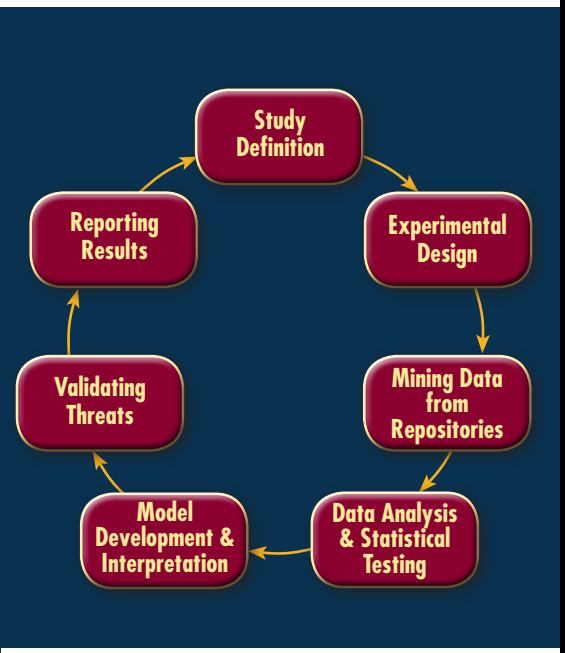
EMPIRICAL RESEARCH IN SOFTWARE ENGINEERING

CONCEPTS, ANALYSIS, AND APPLICATIONS



Ruchika Malhotra

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

EMPIRICAL RESEARCH IN SOFTWARE ENGINEERING

CONCEPTS, ANALYSIS, AND APPLICATIONS

This page intentionally left blank

EMPIRICAL RESEARCH IN SOFTWARE ENGINEERING

CONCEPTS, ANALYSIS, AND APPLICATIONS

Ruchika Malhotra

CRC Press
Taylor & Francis Group
Boca Raton London New York

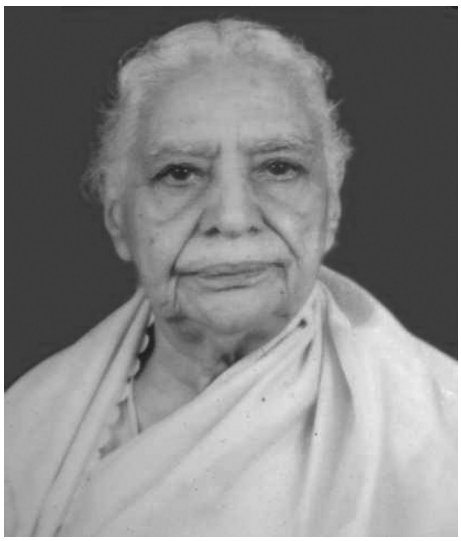
CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 20160120

International Standard Book Number-13: 978-1-4987-1973-5 (eBook - PDF)

ॐ



I dedicate this book to my grandmother, the late Shrimati Shakuntala Rani Malhotra, for her infinite love, understanding, and support. Without her none of my success would have been possible and I would not be who I am today. I miss her very much.

This page intentionally left blank

Contents

This page intentionally left blank

Foreword

Dr. Ruchika Malhotra is a leading researcher in the software engineering community in India, whose work has applications and implications across the broad spectrum of software engineering activities and domains. She is also one of the leading scholars in the area of search-based software engineering (SBSE) within this community, a topic that is rapidly attracting interest and uptake within the wider Indian software engineering research and practitioner communities and is also well established worldwide as a widely applicable approach to software product and process optimization.

In this book Dr. Malhotra uses her breadth of software engineering experience and expertise to provide the reader with coverage of many aspects of empirical software engineering. She covers the essential techniques and concepts needed for a researcher to get started on empirical software engineering research, including metrics, experimental design, analysis and statistical techniques, threats to the validity of any research findings, and methods and tools for empirical software engineering research.

As Dr. Malhotra notes in this book, SBSE is one approach to software engineering that is inherently grounded in empirical assessment, since the overall approach uses computational search to optimize software engineering problems. As such, almost all research work on SBSE involves some form of empirical assessment.

Dr. Malhotra also reviews areas of software engineering that typically rely heavily on empirical techniques, such as software repository mining, an area of empirical software engineering research that offers a rich set of insights into the nature of our engineering material and processes. This is a particularly important topic area that exploits the vast resources of data available (in open source repositories, in app stores, and in other electronic resources relating to software projects). Through mining these repositories, we answer fundamentally empirical questions about software systems that can inform practice and software improvement.

The book provides the reader with an introduction and overview of the field and is also backed by references to the literature, allowing the interested reader to follow up on the methods, tools, and concepts described.

Mark Harman
University College London

This page intentionally left blank

Preface

Empirical research has become an essential component of software engineering research practice. Empirical research in software engineering—including the concepts, analysis, and applications—is all about designing, collecting, analyzing, assessing, and interpreting empirical data collected from software repositories using statistical and machine-learning techniques. Software practitioners and researchers can use the results obtained from these analyses to produce high quality, low cost, and maintainable software.

Empirical software engineering involves planning, designing, analyzing, assessing, interpreting, and reporting results of validation of empirical data. There is a lack of understanding and level of uncertainty on the empirical procedures and practices in software engineering. The aim of this book is to present the empirical research processes, procedures, and practices that can be implemented in practice by the research community. My several years of experience in the area of empirical software engineering motivated me to write this book.

Universities and software industries worldwide have started realizing the importance of empirical software engineering. Many universities are now offering a full course on empirical software engineering for undergraduate, postgraduate, and doctoral students in the disciplines of software engineering, computer science engineering, information technology, and computer applications.

In this book, a description of the steps followed in the research process in order to carry out replicated and empirical research is presented. Readers will gain practical knowledge about how to plan and design experiments, conduct systematic reviews and case studies, and analyze the results produced by these empirical studies. Hence, the empirical research process will provide the software engineering community the knowledge for conducting empirical research in software engineering.

The book contains a judicious mix of empirical research concepts and real-life case study that makes it ideal for a course and research on empirical software engineering. Readers will also experience the process of developing predictive models (e.g., defect prediction, change prediction) on data collected from source code repositories. The purpose of this book is to introduce students, academicians, teachers, software practitioners and researchers to the research process carried out in the empirical studies in software engineering. This book presents the application of machine-learning techniques and real-life case studies in empirical software engineering, which aims to bridge the gap between research and practice. The book explains the concepts with numerous examples of empirical studies. The main features of this book are as follows:

- Presents empirical processes, the importance of empirical-research ethics in software engineering research, and the types of empirical studies.
- Describes the planning, conducting, and reporting phases of systematic literature reviews keeping in view the challenges encountered in this field.

- Describes software metrics; the most popular metrics given to date are included and explained with the help of examples.
- Provides an in-depth description of experimental design, including research questions formation, literature review, variables description, hypothesis formulation, data collection, and selection of data analysis methods.
- Provides a full chapter on mining data from software repositories. It presents the procedure for extracting data from software repositories such as CVS, SVN, and Git and provides applications of the data extracted from these repositories in the software engineering area.
- Describes methods for analyzing data, hypothesis testing, model prediction, and interpreting results. It presents statistical tests with examples to demonstrate their use in the software engineering area.
- Describes performance measures and model validation techniques. The guidelines for using the statistical tests and performance measures are also provided. It also emphasizes the use of machine-learning techniques in predicting models along with the issues involved with these techniques.
- Summarizes the categories of threats to validity with practical examples. A summary of threats extracted from fault prediction studies is presented.
- Provides guidelines to researchers and doctorate students for publishing and reporting the results. Research misconduct is discussed.
- Presents the procedure for mining unstructured data using text mining techniques and describes the concepts with the help of examples and case studies. It signifies the importance of text-mining procedures in extracting relevant information from software repositories and presents the steps in text mining.
- Presents real-life research-based case studies on software quality prediction models. The case studies are developed to demonstrate the procedures used in the chapters of the book.
- Presents an overview of tools that are widely used in the software industry for carrying out empirical studies.

I take immense pleasure in presenting to you this book and hope that it will inspire researchers worldwide to utilize the knowledge and knowhow contained for newer applications and advancement of the frontiers of software engineering. The importance of feedback from readers is important to continuously improve the contents of the book. I welcome constructive criticism and comments about anything in the book; any omission is due to my oversight. I will appreciatively receive suggestions, which will motivate me to work hard on a next improved edition of the book; as Robert Frost rightly wrote:

> The woods are lovely, dark, and deep,
> But I have promises to keep,
> And miles to go before I sleep,
> And miles to go before I sleep.

Ruchika Malhotra
Delhi Technological University

Acknowledgments

I am most thankful to my father for constantly encouraging me and giving me time and unconditional support while writing this book. He has been a major source of inspiration to me.

This book is a result of the motivation of Yogesh Singh, Director, Netaji Subhas Institute of Technology, Delhi, India. It was his idea that triggered this book. He has been a constant source of inspiration for me throughout my research and teaching career. He has been continuously involved in evaluating and improving the chapters in this book; I will always be indebted to him for his extensive support and guidance.

I am extremely grateful to Mark Harman, professor of software engineering and head of the Software Systems Engineering Group, University College London, UK, for the support he has given to the book in his foreword. He has contributed greatly to the field of software engineering research and was the first to explore the use and relevance of search-based techniques in software engineering.

I am extremely grateful to Megha Khanna, assistant professor, Acharya Narendera Dev College, University of Delhi, India, for constantly working with me in terms of solving examples, preparing case studies, and reading texts. The book would not have been in its current form without her support. My sincere thanks to her.

My heartfelt gratitude is due to Ankita Jain Bansal, assistant professor, Netaji Subhas Institute of Technology, Delhi, India, who worked closely with me during the evolution of the book. She was continuously involved in modifying various portions in the book, especially experimental design procedure and threshold analysis.

I am grateful to Abha Jain, research scholar, Delhi Technological University, Delhi, India, for helping me develop performance measures and text-mining examples for the book. My thanks are also due to Kanishk Nagpal, software engineer, Samsung India Electronics Limited, Delhi, India, who worked closely with me on mining repositories and who developed the DCRS tool provided in Chapter 5.

Thanks are due to all my doctoral students in the Department of Software Engineering, Delhi Technological University, Delhi, India, for motivating me to explore and evolve empirical research concepts and applications in software engineering. Thanks are also due to my undergraduate and postgraduate students at the Department of Computer Science and Software Engineering, Delhi Technological University, for motivating me to study more before delivering lectures and exploring and developing various tools in several projects. Their outlook, debates, and interest have been my main motivation for continuous advancement in my academic pursuit. I also thank all researchers, scientists, practitioners, software developers, and teachers whose insights, opinions, ideas, and techniques find a place in this book.

Thanks also to Rajeev Raje, professor, Department of Computer & Information Science, Indiana University–Purdue University Indianapolis, Indianapolis, for his support and valuable suggestions.

Last, I am thankful to Manju Khari, assistant professor, Ambedkar Institute of Technology, Delhi, India, for her support in gathering some of the material for the further readings sections in the book.

This page intentionally left blank

Author

Ruchika Malhotra is an assistant professor in the Department of Software Engineering, Delhi Technological University (formerly Delhi College of Engineering), Delhi, India. She is a Raman Scholar and was awarded the prestigious UGC Raman Postdoctoral Fellowship by the government of India, under which she pursued postdoctoral research in the Department of Computer and Information Science, Indiana University–Purdue University Indianapolis (2014–15), Indiana. She earned her master's and doctorate degrees in software engineering from the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. She was an assistant professor at the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India.

Dr. Malhotra received the prestigious IBM Faculty Award in 2013 and has received the Best Presenter Award in Workshop on Search Based Software Testing, ICSE, 2014, Hyderabad, India. She is an executive editor of *Software Engineering: An International Journal* and is a coauthor of the book, *Object Oriented Software Engineering*.

Dr. Malhotra's research interests are in empirical research in software engineering, improving software quality, statistical and adaptive prediction models, software metrics, the definition and validation of software metrics, and software testing. Her *H*-index as reported by Google Scholar is 17. She has published more than 100 research papers in international journals and conferences, and has been a referee for various journals of international repute in the areas of software engineering and allied fields. She is guiding several doctoral candidates and has guided several undergraduate projects and graduate dissertations. She has visited foreign universities such as Imperial College, London, UK; Indiana University–Purdue University Indianapolis, Indiana; Ball State University, Muncie, Indiana; and Harare Institute of Technology, Zimbabwe. She has served on the technical committees of several international conferences in the area of software engineering (SEED, WCI, ISCON). Dr. Malhotra can be contacted via e-mail at ruchikamalhotra2004@yahoo.com.

This page intentionally left blank

1

Introduction

As the size and complexity of software is increasing, software organizations are facing the pressure of delivering high-quality software within a specific time, budget, and available resources. The software development life cycle consists of a series of phases, including requirements analysis, design, implementation, testing, integration, and maintenance. Software professionals want to know which tools to use at each phase in software development and desire effective allocation of available resources. The software planning team attempts to estimate the cost and duration of software development, the software testers want to identify the fault-prone modules, and the software managers seek to know which tools and techniques can be used to reduce the delivery time and best utilize the manpower. In addition, the software managers also desire to improve the software processes so that the quality of the software can be enhanced. Traditionally, the software engineers have been making decisions based on their intuition or individual expertise without any scientific evidence or support on the benefits of a tool or a technique.

Empirical studies are verified by observation or experiment and can provide powerful evidence for testing a given hypothesis (Aggarwal et al. 2009). Like other disciplines, software engineering has to adopt empirical methods that will help to plan, evaluate, assess, monitor, control, predict, manage, and improve the way in which software products are produced. An empirical study of real systems can help software organizations assess large software systems quickly, at low costs. The application of empirical techniques is especially beneficial for large-scale systems, where software professionals need to focus their attention and resources on various activities of the system under development. For example, developing a model for predicting faulty modules allows software organizations to identify faulty portions of source code so that testing activities can be planned more effectively. Empirical studies such as surveys, systematic reviews and experimental studies, help software practitioners to scientifically assess and validate the tools and techniques in software development.

In this chapter, an overview and the types of empirical studies are provided, the phases of the experimental process are described, and the ethics involved in empirical research of software engineering are summarized. Further, this chapter also discusses the key concepts used in the book.

1.1 What Is Empirical Software Engineering?

The initial debate on software as an engineering discipline is over now. It has been realized that without software as an engineering discipline survival is difficult. Engineering compels the development of the product in a scientific, well formed, and systematic manner. Core engineering principles should be applied to produce good quality maintainable software

within a specified time and budget. Fritz Bauer coined the term *software engineering* in 1968 at the first conference on software engineering and defined it as (Naur and Randell 1969):

> The establishment and use of sound engineering principles in order to obtain economically developed software that is reliable and works efficiently on real machines.

Software engineering is defined by IEEE Computer Society as (Abren et al. 2004):

> The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software, and the study of these approaches, that is, the application of engineering to software.

The software engineering discipline facilitates the completion of the objective of delivering good quality software to the customer following a systematic and scientific approach. Empirical methods can be used in software engineering to provide scientific evidence on the use of tools and techniques.

Harman et al. (2012a) defined "empirical" as:

> "Empirical" is typically used to define any statement about the world that is related to observation or experience.

Empirical software engineering (ESE) is an area of research that emphasizes the use of empirical methods in the field of software engineering. It involves methods for evaluating, assessing, predicting, monitoring, and controlling the existing artifacts of software development.

ESE applies quantitative methods to the software engineering phenomenon to understand software development better. ESE has been gaining importance over the past few decades because of the availability of vast data sets from open source repositories that contain information about software requirements, bugs, and changes (Meyer et al. 2013).

1.2 Overview of Empirical Studies

Empirical study is an attempt to compare the theories and observations using real-life data for analysis. Empirical studies usually utilize data analysis methods and statistical techniques for exploring relationships. They play an important role in software engineering research by helping to form well-formed theories and widely accepted results. The empirical studies provide the following benefits:

- Allow to explore relationships
- Allow to prove theoretical concepts
- Allow to evaluate accuracy of the models
- Allow to choose among tools and techniques
- Allow to establish quality benchmarks across software organizations
- Allow to assess and improve techniques and methods

Empirical studies are important in the area of software engineering as they allow software professionals to evaluate and assess the new concepts, technologies, tools, and techniques in scientific and proved manner. They also allow improving, managing, and controlling the existing processes and techniques by using evidence obtained from the empirical analysis. The empirical information can help software management in decision making

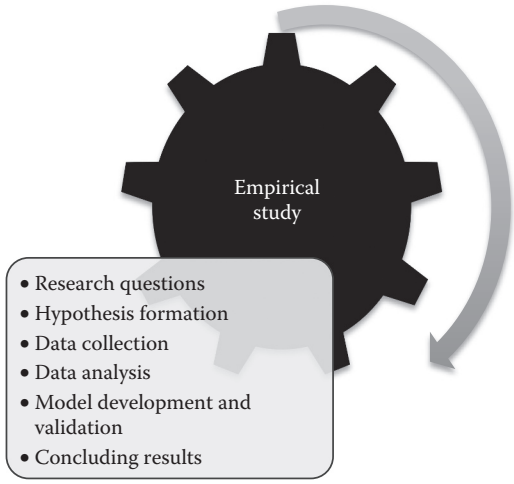


FIGURE 1.1
Steps in empirical studies.

and improving software processes. The empirical studies involve the following steps (Figure 1.1):

- Formation of research questions
- Formation of a research hypothesis
- Gathering data
- Analyzing the data
- Developing and validating models
- Deriving conclusions from the obtained results

Empirical study allows to gather evidence that can be used to support the claims of efficiency of a given technique or technology. Thus, empirical studies help in building a body of knowledge so that the processes and products are improved resulting in high-quality software.

Empirical studies are of many types, including surveys, systematic reviews, experiments, and case studies.

1.3 Types of Empirical Studies

The studies can be broadly classified as quantitative and qualitative. Quantitative research is the most widely used scientific method in software engineering that applies mathematical- or statistical-based methods to derive conclusions. Quantitative research is used to prove or disprove a hypothesis (a concept that has to be tested for further investigation). The aim of a quantitative research is to generate results that are generalizable and unbiased and thus can be applied to a larger population in research. It uses statistical methods to validate a hypothesis and to explore causal relationships.

In qualitative research, the researchers study human behavior, preferences, and nature. Qualitative research provides an in-depth analysis of the concept under investigation and thus uses focused data for research. Understanding a new process or technique in software engineering is an example of qualitative research. Qualitative research provides textual descriptions or pictures related to human beliefs or behavior. It can be extended to other studies with similar populations but generalizations of a particular phenomenon may be difficult. Qualitative research involves methods such as observations, interviews, and group discussions. This method is widely used in case studies.

Qualitative research can be used to analyze and interpret the meaning of results produced by quantitative research. Quantitative research generates numerical data for analysis, whereas qualitative research generates non-numerical data (Creswell 1994). The data of qualitative research is quite rich as compared to quantitative data. Table 1.1 summaries the key differences between quantitative and qualitative research.

The empirical studies can be further categorized as experimental, case study, systematic review, survey, and post-mortem analysis. These categories are explained in the next section. Figure 1.2 presents the quantitative and qualitative types of empirical studies.

1.3.1 Experiment

An experimental study tests the established hypothesis by finding the effect of variables of interest (independent variables) on the outcome variable (dependent variable) using statistical analysis. If the experiment is carried out correctly, the hypothesis is either accepted or rejected. For example, one group uses technique A and the other group uses technique B, which technique is more effective in detecting a larger number of defects? The researcher may apply statistical tests to answer such questions. According to Kitchenham et al. (1995), the experiments are small scale and must be controlled. The experiment must also control the confounding variables, which may affect the accuracy of the results produced by the experiment. The experiments are carried out in a controlled environment and often referred to as controlled experiments (Wohlin 2012).

The key factors involved in the experiments are independent variables, dependent variables, hypothesis, and statistical techniques. The basic steps followed in experimental

TABLE 1.1

Comparison of Quantitative and Qualitative Research

	Quantitative Research	Qualitative Research
General	Objective	Subjective
Concept	Tests theory	Forms theory
Focus	Testing a hypothesis	Examining the depth of a phenomenon
Data type	Numerical	Textual or pictorial
Group	Small	Large and random
Purpose	Predict causal relationships	Describe and interpret concepts
Basis	Based on hypothesis	Based on concept or theory
Method	Confirmatory: established hypothesis is tested	Exploratory: new hypothesis is formed
Variables	Variables are defined by the researchers	Variables may emerge unexpectedly
Settings	Controlled	Flexible
Results	Generalizable	Specialized

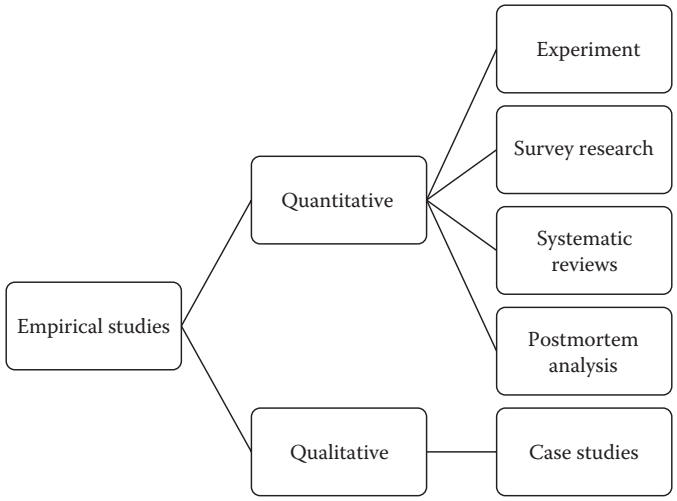


FIGURE 1.2
Types of empirical studies.

FIGURE 1.3
Steps in experimental research.

research are shown in Figure 1.3. The same steps are followed in any empirical study process however the content varies according to the specific study being carried out. In first phase, experiment is defined. The next phase involves determining the experiment design. In the third phase the experiment is executed as per the experiment design. Then, the results are interpreted. Finally, the results are presented in the form of experiment report. To carry out an empirical study, a replicated study (repeating a study with similar settings or methods but different data sets or subjects), or to perform a survey of existing empirical studies, the research methodology followed in these studies needs to be formulated and described.

A controlled experiment involves varying the variables (one or more) and keeping everything else constant or the same and are usually conducted in small or laboratory setting (Conradi and Wang 2003). Comparing two methods for defect detection is an example of a controlled experiment in software engineering context.

1.3.2 Case Study

Case study research represents real-world scenarios and involves studying a particular phenomenon (Yin 2002). Case study research allows software industries to evaluate a tool,

method, or process (Kitchenham et al. 1995). The effect of a change in an organization can be studied using case study research. Case studies increase the understanding of the phenomenon under study. For example, a case study can be used to examine whether a unified model language (UML) tool is effective for a given project or not. The initial and new concepts are analyzed and explored by exploratory case studies, whereas the already existing concepts are tested and improvised by confirmatory case studies.

The phases included in the case study are presented in Figure 1.4. The case study design phase involves identifying existing objectives, cases, research questions, and data-collection strategies. The case may be a tool, technology, technique, process, product, individual, or software. Qualitative data is usually collected in a case study. The sources include interviews, group discussions, or observations. The data may be directly or indirectly collected from participants. Finally, the case study is executed, the results obtained are analyzed, and the findings are reported. The report type may vary according to the target audience.

Case studies are appropriate where a phenomenon is to be studied for a longer period of time so that the effects of the phenomenon can be observed. The disadvantages of case studies include difficulty in generalization as they represent a typical situation. Since they are based on a particular case, the validity of the results is questionable.

1.3.3 Survey Research

Survey research identifies features or information from a large scale of a population. For example, surveys can be used when a researcher wants to know whether the use of a particular process has improved the view of clients toward software usability features. This information can be obtained by asking the selected software testers to fill questionnaires. Surveys are usually conducted using questionnaires and interviews. The questionnaires are constructed to collect research-related information.

Preparation of a questionnaire is an important activity and should take into consideration the features of the research. The effective way to obtain a participant's opinion is to get a questionnaire or survey filled by the participant. The participant's feedback and reactions are recorded in the questionnaire (Singh 2011). The questionnaire/survey can be used to detect trends and may provide valuable information and feedback on a particular process, technique, or tool. The questionnaire/survey must include questions concerning the participant's likes and dislikes about a particular process, technique, or tool. The interviewer should preferably handle the questionnaire.

Surveys are classified into three types (Babbie 1990)—descriptive, explorative, and explanatory. Exploratory surveys focus on the discovery of new ideas and insights and are usually conducted at the beginning of a research study to gather initial information. The descriptive survey research is more detailed and describes a concept or topic. Explanatory survey research tries to explain how things work in connections like cause and effect, meaning a researcher wants to explain how things interact or work with each other. For example, while exploring relationship between various independent variables and an

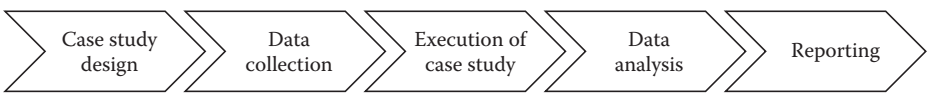


FIGURE 1.4
Case study phases.

outcome variable, a researcher may want to explain why an independent variable affects the outcome variable.

1.3.4 Systematic Reviews

While conducting any study, literature review is an important part that examines the existing position of literature in an area in which the research is being conducted. The systematic reviews are methodically undertaken with a specific search strategy and well-defined methodology to answer a set of questions. The aim of a systematic review is to analyze, assess, and interpret existing results of research to answer research questions. Kitchenham (2007) defines systematic review as:

> A form of secondary study that uses a well-defined methodology to identify, analyze and interpret all available evidence related to a specific research question in a way that is unbiased and (to a degree) repeatable.

The purpose of a systematic review is to summarize the existing research and provide future guidelines for research by identifying gaps in the existing literature. A systematic review involves:

1. Defining research questions.
2. Forming and documenting a search strategy.
3. Determining inclusion and exclusion criteria.
4. Establishing quality assessment criteria.

The systematic reviews are performed in three phases: planning the review, conducting the review, and reporting the results of the review. Figure 1.5 presents the summary of the phases involved in systematic reviews.

In the planning stage, the review protocol is developed that includes the following steps: research questions identification, development of review protocol, and evaluation of review protocol. During the development of review protocol the basic processes in the review are planned. The research questions are formed that address the issues to be

FIGURE 1.5
Phases of systematic review.

answered in the systematic literature review. The development of review protocol involves planning a series of steps—search strategy design, study selection criteria, study quality assessment, data extraction process, and data synthesis process. In the first step, the search strategy is described that includes identification of search terms and selection of sources to be searched to identify the primary studies. The second step determines the inclusion and exclusion criteria for each primary study. In the next step, the quality assessment criterion is identified by forming the quality assessment questionnaire to analyze and assess the studies. The second to last step involves the design of data extraction forms to collect the required information to answer the research questions, and, in the last step, data synthesis process is defined. The above series of steps are executed in the review in the conducting phase. In the final phase, the results are documented. Chapter 2 provides details of systematic review.

1.3.5 Postmortem Analysis

Postmortem analysis is carried out after an activity or a project has been completed. The main aim is to detect how the activities or processes can be improved in the future. The postmortem analysis captures knowledge from the past, after the activity has been completed. Postmortem analysis can be classified into two types: general postmortem analysis and focused postmortem analysis. General postmortem analysis collects all available information from a completed activity, whereas focused postmortem analysis collects information about specific activity such as effort estimation (Birk et al. 2002).

According to Birk et al., in postmortem analysis, large software systems are analyzed to gain knowledge about the good and bad practices of the past. The techniques such as interviews and group discussions can be used for collecting data in postmortem analysis. In the analysis process, the feedback sessions are conducted where the participants are asked whether the concepts told to them have been correctly understood (Birk et al. 2002).

1.4 Empirical Study Process

Before describing the steps involved in the empirical research process, it is important to distinguish between empirical and experimental approaches as they are often used interchangeably but are slightly different from each other. Harman et al. (2012a) makes a distinction between experimental and empirical approaches in software engineering. In experimental software engineering, the dependent variable is closely observed in a controlled environment. Empirical studies are used to define anything related to observation and experience and are valuable as these studies consider real-world data. In experimental studies, data is artificial or synthetic but is more controlled. For example, using 5000 machine-generated instances is an experimental study, and using 20 real-world programs in the study is an empirical study (Meyer et al. 2013). Hence, any experimental approach, under controlled environments, allows the researcher to remove the research bias and confounding effects (Harman et al. 2012a). Both empirical and experimental approaches can be combined in the studies.

Without a sound and proven research process, it is difficult to carry out efficient and effective research. Thus, a research methodology must be complete and repeatable, which, when followed, in a replicated or empirical study, will enable comparisons to be made across various studies. Figure 1.6 depicts the five phases in the empirical study process. These phases are discussed in the subsequent subsections.

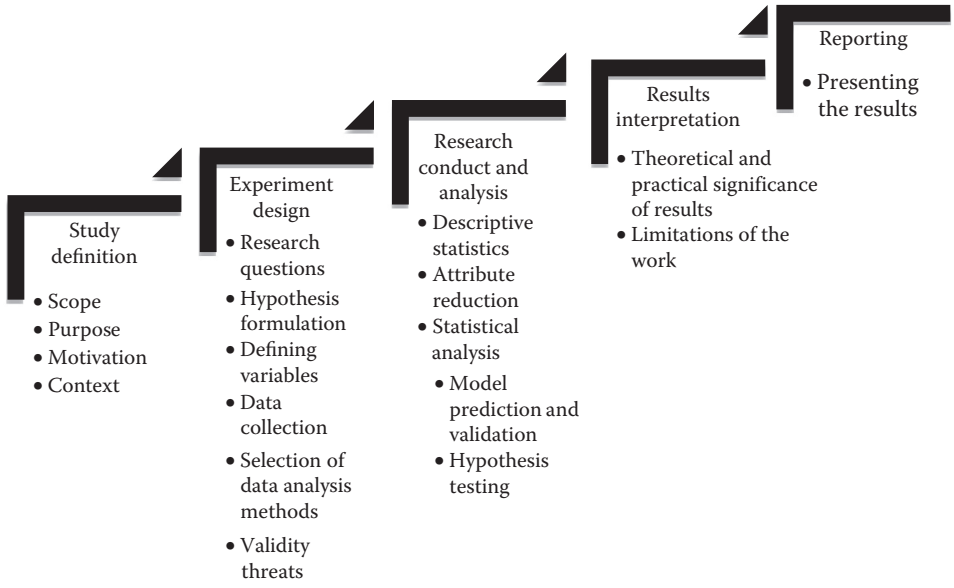


FIGURE 1.6
Empirical study phases.

1.4.1 Study Definition

The first step involves the definition of the goals and objectives of the empirical study. The aim of the study is explained in this step. Basili et al. (1986) suggests dividing the defining phase into the following parts:

- Scope: What are the dimensions of the study?
- Motivation: Why is it being studied?
- Object: What entity is being studied?
- Purpose: What is the aim of the study?
- Perspective: From whose view is the study being conducted (e.g, project manager, customer)?
- Domain: What is the area of the study?

The scope of the empirical study defines the extent of the investigation. It involves listing down the specific goals and objectives of the experiment. The purpose of the study may be to find the effect of a set of variables on the outcome variable or to prove that technique A is superior to technique B. It also involves identifying the underlying hypothesis that is formulated at later stages. The motivation of the experiment describes the reason for conducting the study. For example, the motivation of the empirical study is to analyze and assess the capability of a technique or method. The object of the study is the entity being examined in the study. The entity in the study may be the process, product, or technique. Perspective defines the view from which the study is conducted. For example, if the study is conducted from the tester's point of view then the tester will be interested in planning and allocating resources to test faulty portions of the source code. Two important domains in the study are programmers and programs (Basili et al. 1986).

1.4.2 Experiment Design

This is the most important and significant phase in the empirical study process. The design of the experiment covers stating the research questions, formation of the hypothesis, selection of variables, data-collection strategies, and selection of data analysis methods. The context of the study is defined in this phase. Thus, the sources (university/academic, industrial, or open source) from which the data will be collected are identified. The data-collection process must be well defined and the characteristics of the data must be stated. For example, nature, programming language, size, and so on must be provided. The outcome variables are to be carefully selected such that the objectives of the research are justified. The aim of the design phase should be to select methods and techniques that promote replicability and reduce experiment bias (Pfleeger 1995). Hence, the techniques used must be clearly defined and the settings should be stated so that the results can be replicated and adopted by the industry. The following are the steps carried out during the design phase:

1. Research questions: The first step is to formulate the research problem. This step states the problem in the form of questions and identifies the main concepts and relations to be explored. For example, the following questions may be addressed in empirical studies to find the relationship between software metrics and quality attributes:
   a. What will be the effect of software metrics on quality attributes (such as fault proneness/testing effort/maintenance effort) of a class?
   b. Are machine-learning methods adaptable to object-oriented systems for predicting quality attributes?
   c. What will be the effect of software metrics on fault proneness when severity of faults is taken into account?
2. Independent and dependent variables: To analyze relationships, the next step is to define the dependent and the independent variables. The outcome variable predicted by the independent variables is called the dependent variable. For instance, the dependent variables of the models chosen for analysis may be fault proneness, testing effort, and maintenance effort. A variable used to predict or estimate a dependent variable is called the independent (explanatory) variable.
3. Hypothesis formulation: The researcher should carefully state the hypothesis to be tested in the study. The hypothesis is tested on the sample data. On the basis of the result from the sample, a decision concerning the validity of the hypothesis (acception or rejection) is made.

   Consider an example where a hypothesis is to be formed for comparing a number of methods for predicting fault-prone classes.

   For each method, *M*, the hypothesis in a given study is the following (the relevant null hypothesis is given in parentheses), where the capital *H* indicates "hypothesis." For example:

   H–M: *M* outperform the compared methods for predicting fault-prone software classes (null hypothesis: *M* does not outperform the compared methods for predicting fault-prone software classes).
4. Empirical data collection: The researcher decides the sources from which the data is to be collected. It is found from literature that the data collected is either from university/academic systems, commercial systems, or open source software. The researcher should state the environment in which the study is performed,

programming language in which the systems are developed, size of the systems to be analyzed (lines of code [LOC] and number of classes), and the duration for which the system is developed.

5. Empirical methods: The data analysis techniques are selected based on the type of the dependent variables used. An appropriate data analysis technique should be selected by identifying its strengths and weaknesses. For example, a number of techniques have been available for developing models to predict and analyze software quality attributes. These techniques could be statistical like linear regression and logistic regression or machine-learning techniques like decision trees, support vector machines, and so on. Apart from these techniques, there are a new set of techniques like particle swarm optimization, gene expression programming, and so on that are called the search-based techniques. The details of these techniques can be found in Chapter 7.

In the empirical study, the data is analyzed corresponding to the details given in the experimental design. Thus, the experimental design phase must be carefully planned and executed so that the analysis phase is clear and unambiguous. If the design phase does not match the analysis part then it is most likely that the results produced are incorrect.

1.4.3 Research Conduct and Analysis

Finally, the empirical study is conducted following the steps described in the experiment design. The experiment analysis phase involves understanding the data by collecting descriptive statistics. The unrelated attributes are removed, and the best attributes (variables) are selected out of a set of attributes (e.g., software metrics) using attribute reduction techniques. After removing irrelevant attributes, hypothesis testing is performed using statistical tests and, on the basis of the result obtained, a decision regarding the acceptance or rebuttal of the hypothesis is made. The statistical tests are described in Chapter 6. Finally, for analyzing the casual relationships between the independent variables and the dependent variable, the model is developed and validated. The steps involved in experiment conduct and analysis are briefly described below.

1. Descriptive statistics: The data is validated for correctness before carrying out the analysis. The first step in the analysis is descriptive statistics. The research data must be suitably reduced so that the research data can be read easily and can be used for further analysis. Descriptive statistics concern development of certain indices or measures to summarize the data. The important statistics measures used for comparing different case studies include mean, median, and standard deviation. The data analysis methods are selected based on the type of the dependent variable being used. Statistical tests can be applied to accept or refute a hypothesis. Significance tests are performed for comparing the predicted performance of a method with other sets of methods. Moreover, effective data assessment should also yield outliers (Aggarwal et al. 2009).
2. Attribute reduction: Feature subselection is an important step that identifies and removes as much of the irrelevant and redundant information as possible. The dimensionality of the data reduces the size of the hypothesis space and allows the methods to operate faster and more effectively (Hall 2000).
3. Statistical analysis: The data collected can be analyzed using statistical analysis by following the steps below.

a. Model prediction: The multivariate analysis is used for the model prediction. Multivariate analysis is used to find the combined effect of each independent variable on the dependent variable. Based on the results of performance measures, the performance of models predicted is evaluated and the results are interpreted. Chapter 7 describes these performance measures.
b. Model validation: In systems, where models are independently constructed from the training data (such as in data mining), the process of constructing the model is called training. The subsamples of data that are used to validate the initial analysis (by acting as "blind" data) are called validation data or test data. The validation data is used for validating the model predicted in the previous step.
c. Hypothesis testing: It determines whether the null hypothesis can be rejected at a specified confidence level. The confidence level is determined by the researcher and is usually less than 0.01 or 0.05 (refer Section 4.7 for details).

1.4.4 Results Interpretation

In this step, the results computed in the empirical study's analysis phase are assessed and discussed. The reason behind the acceptance or rejection of the hypothesis is examined. This process provides insight to the researchers about the actual reasons of the decision made for hypothesis. The conclusions are derived from the results obtained in the study. The significance and practical relevance of the results are defined in this phase. The limitations of the study are also reported in the form of threats to validity.

1.4.5 Reporting

Finally, after the empirical study has been conducted and interpreted, the study is reported in the desired format. The results of the study can be disseminated in the form of a conference article, a journal paper, or a technical report.

The results are to be reported from the reader's perspective. Thus, the background, motivation, analysis, design, results, and the discussion of the results must be clearly documented. The audience may want to replicate or repeat the results of a study in a similar context. The experiment settings, data-collection methods, and design processes must be reported in significant level of detail. For example, the descriptive statistics, statistical tools, and parameter settings of techniques must be provided. In addition, graphical representation should be used to represent the results. The results may be graphically presented using pie charts, line graphs, box plots, and scatter plots.

1.4.6 Characteristics of a Good Empirical Study

The characteristics of a good empirical study are as follows:

1. Clear: The research goals, hypothesis, and data-collection procedure must be clearly stated.
2. Descriptive: The research should provide details of the experiment so that the study can be repeated and replicated in similar settings.
3. Precise: Precision helps to prove confidence in the data. It represents the degree of measure correctness and data exactness. High precision is necessary to specify the attributes in detail.

4. Valid: The experiment conclusions should be valid for a wide range of population.
5. Unbiased: The researcher performing the study should not influence the results to satisfy the hypothesis. The research may produce some bias because of experiment error. The bias may be produced when the researcher selects the participants such that they generate the desired results. The measurement bias may occur during data collection.
6. Control: The experiment design should be able to control the independent variables so that the confounding effects (interaction effects) of variables can be reduced.
7. Replicable: Replication involves repeating the experiment with different data under same experimental conditions. If the replication is successful then this indicates generalizability and validity of the results.
8. Repeatable: The experimenter should be able to reproduce the results of the study under similar settings.

1.5 Ethics of Empirical Research

Researchers, academicians, and sponsors should be aware of research ethics while conducting and funding empirical research in software engineering. The upholding of ethical standards helps to develop trust between the researcher and the participant, and thus smoothens the research process. An unethical study can harm the reputation of the research conducted in software engineering area.

Some ethical issues are regulated by the standards and laws provided by the government. In some countries like the United States, the sponsoring agency requires that the research involving participants must be reviewed by a third-party ethics committee to verify that the research complies with the ethical principles and standards (Singer and Vinson 2001). Empirical research is based on the trust between the participant and the researcher, the ethical information must be explicitly provided to the participants to avoid any future conflicts. The participants must be informed about the risk and ethical issues involved in the research at the beginning of the study. The examples of problems related to ethics that are experienced in industry are given by Becker-Kornstaedt (2001) and summarized in Table 1.2.

TABLE 1.2

Examples of Unethical Research

S. No	Problem
1	Employees misleading the manager to protect himself or herself with the knowledge of the researcher
2	Nonconformance to a mandatory process
3	Revealing identities of the participant or organization
4	Manager unexpectedly joining a group interview or discussion with the participant
5	Experiment revealing identity of the participants of a nonperforming department in an organization
6	Experiment outcomes are used in employee ratings
7	Participants providing information off the record, that is, after interview or discussion is over

The ethical threats presented in Table 1.2 can be reduced by (1) presenting data and results such that no information about the participant and the organization is revealed, (2) presenting different reports to stakeholders, (3) providing findings to the participants and giving them the right to withdraw any time during the research, and (4) providing publication to companies for review before being published. Singer and Vinson (2001) identified that the engineering and science ethics may not be related to empirical research in software engineering. They provided the following four ethical principles:

1. Informed consent: This principle is concerned with subjects participating in the experiment. The subjects or participants must be provided all the relevant information related to the experiment or study. The participants must willingly agree to participate in the research process. The consent form acts as a contract between an individual participant and the researcher.
2. Scientific value: This principle states that the research results must be correct and valid. This issue is critical if the researchers are not familiar with the technology or methodology they are using as it will produce results of no scientific value.
3. Confidentiality: It refers to anonymity of data, participants, and organizations.
4. Beneficence: The research must provide maximum benefits to the participants and protect the interests of the participants. The benefits of the organization must also be protected by not revealing the weak processes and procedures being followed in the departments of the organization.

1.5.1 Informed Content

Informed consent consists of five elements—disclosure, comprehension, voluntariness, consent, and right to withdraw. Disclosure means to provide all relevant details about the research to the participants. This information includes risks and benefits incurred by the participants. Comprehension refers to presenting information in such a manner that can be understood by the participant. Voluntariness specifies that the consent obtained must not be under any pressure or influence and actual consent must be taken. Finally, the subjects must have the right to withdraw from research process at any time. The consent form has the following format (Vinson and Singer 2008):

1. Research title: The title of the project must be included in the consent form.
2. Contact details: The contact details (including ethics contact) will provide the participant information about whom to contact to clarify any questions or issues or complaints.
3. Consent and comprehension: The participant actually gives the consent form in this section stating that they have understood the requirement of the research.
4. Withdrawal: This section states that the participants can withdraw from the research without any penalty.
5. Confidentiality: It states the confidentiality related to the research study.
6. Risks and benefits: This section states the risks and benefits of the research to the participants.
7. Clarification: The participants can ask for any further clarification at any time during the research.
8. Signature: Finally, the participant signs the consent form with the date.

1.5.2 Scientific Value

This ethical issue is concerned with two aspects—relevance of research topic and validity of research results. The research must balance between risks and benefits. In fact, the advantages of the research should outweigh the risks. The results of the research must also be valid. If they are not valid then the results are incorrect and the study has no value to the research community. The reason for invalid results is usually misuse of methodology, application, or tool. Hence, the researchers should not conduct the research for which they are not capable or competent.

1.5.3 Confidentiality

The information shared by the participants should be kept confidential. The researcher should hide the identity of the organization and participant. Vinson and Singer (2008) identified three features of confidentiality—data privacy, participant anonymity, and data anonymity. The data collected must be protected by password and only the people involved in the research should have access to it. The data should not reveal the information about the participant. The researchers should not collect personal information of participant. For example, participant identity must be used instead of the participant name. The participant information hiding is achieved by hiding information from colleagues, professors, and general public. Hiding information from the manager is particularly essential as it may affect the career of the participants. The information must be also hidden from the organization's competitors.

1.5.4 Beneficence

The participants must be benefited by the research. Hence, methods that protect the interest of the participants and do not harm them must be adopted. The research must not pose a threat to the researcher's job, for example, by creating an employee-ranking framework. The revealing of an organization's sensitive information may also bring loss to the company in terms of reputation and clients. For example, if the names of companies are revealed in the publication, the comparison between the processes followed in the companies or potential flaws in the processes followed may affect obtaining contracts from the clients. If the research involves analyzing the process of the organization, the outcome of the research or facts revealed from the research can harm the participants to a significant level.

1.5.5 Ethics and Open Source Software

In the absence of empirical data, data and source code from open source software are being widely used for analysis in research. This poses concerns of ethics, as the open source software is not primarily developed for research purposes. El-Emam (2001) raised two important ethical issues while using open source software namely "informed consent and minimization of harm and confidentiality." Conducting studies that rate the developers or compares two open source software may harm the developer's reputation or the company's reputation (El-Emam 2001).

1.5.6 Concluding Remarks

The researcher must maintain the ethics in the research by careful planning and, if required, consulting ethical bodies that have expertise for guiding them on ethical issues in software engineering empirical research. The main aim of the research involving

participants must be to protect the interests of the participants so that they are protected from any harm. Becker-Kornstaedt (2001) suggests that the participant interests can be protected by using techniques such as manipulating data, providing different reports to different stakeholders, and providing the right to withdraw to the participants.

Finally, feedback of the research results must be provided to the participants. The opinion of the participants about the validity of the results must also be asked. This will help in increasing the trust between the researcher and the participant.

1.6 Importance of Empirical Research

Why should empirical studies in software engineering be carried out? The main reason of carrying out an empirical study is to reduce the gap between theory and practice by using statistical tests to test the formed hypothesis. This will help in analyzing, assessing, and improving the processes and procedures of software development. It may also provide guidelines to management for decision making. Thus, without evaluating and assessing new methods, tools, and techniques, their use will be random and effectiveness will be uncertain. The empirical study is useful to researchers, academicians, and the software industry from different perspectives.

1.6.1 Software Industry

The results of ESE must be adopted by the industry. ESE can be used to answer the questions related to practices in industry and can improve the processes and procedures of software development. To match the requirements of the industry, the researcher must ask the following questions while conducting research:

- How does the research aim maps to the industrial problems?
- How can the software practitioners use the research results?
- What are the important problems in the industry?

The predictive models constructed in ESE can be applied to future, similar industrial applications. The empirical research enables software practitioners to use the results of the experiment and ascertain that a set of good processes and procedures are followed during software development. Thus, the empirical study can guide toward determining the quality of the resultant software products and processes. For example, a new technique or technology can be evaluated and assessed. The empirical study can help the software professionals in effectively planning and allocating resources in the initial phases of software development life cycle.

1.6.2 Academicians

While studying or conducting research, academicians are always curious to answer questions that are foremost in their minds. As the academicians dig deeper into their subject or research, the questions tend to become more complex. Empirical research empowers them with a great tool to find an answer by asking or interviewing different stakeholders,

by conducting a survey, or by conducting a scientific experiment. Academicians generally make predictions that can be stated in the form of hypotheses. These hypotheses need to be subjected to robust scientific verification or approval. With empirical research, these hypotheses can be tested, and their results can be stated as either being accepted or rejected. Thereafter, based on the result, the academicians can make some generalization or make a conclusion about a particular theory. In other words, a new theory can be generated and some old ones may be disproved. Additionally, sometimes there are practical questions that an academician encounters, empirical research would be highly beneficial in solving them. For example, an academician working in a university may want to find out the most efficient learning approach that yields the best performance among a group of students. The results of the research can be included in the course curricula.

From the academic point of view, high-quality teaching is important for future software engineers. Empirical research can provide management with important information about the use of tools and techniques. The students will further carry forward the knowledge to the software industry and thus improve the industrial practices. The empirical result can support one technique over the other and hence will be very useful in comparing the techniques.

1.6.3 Researchers

From the researchers point of view, the results can be used to provide insight about existing trends and guidelines regarding future research. The empirical study can be repeated or replicated by the researcher in order to establish generalizability of the results to new subjects or data sets.

1.7 Basic Elements of Empirical Research

The basic elements in empirical research are purpose, participants, process, and product. Figure 1.7 presents the four basic elements of empirical research. The purpose defines the reason of the research, the relevance of the topic, specific aims in the form of research questions, and objectives of the research.

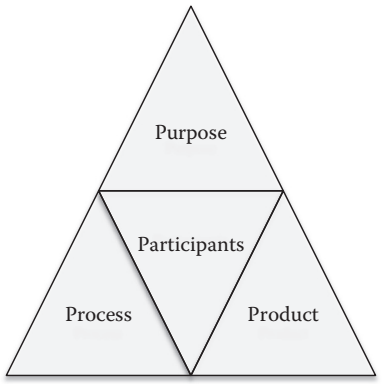


FIGURE 1.7
Elements of empirical research.

Process lays down the way in which the research will be conducted. It defines the sequence of steps taken to conduct a research. It provides details about the techniques, methodologies, and procedures to be used in the research. The data-collection steps, variables involved, techniques applied, and limitations of the study are defined in this step. The process should be followed systematically to produce a successful research.

Participants are the subjects involved in the research. The participants may be interviewed or closely observed to obtain the research results. While dealing with participants, ethical issues in ESE must be considered so that the participants are not harmed in any way.

Product is the outcome produced by the research. The final outcome provides the answer to research questions in the empirical research. The new technique developed or methodology produced can also be considered as a product of the research. The journal paper, conference article, technical report, thesis, and book chapters are products of the research.

1.8 Some Terminologies

Some terminologies that are frequently used in the empirical research in software engineering are discussed in this section.

1.8.1 Software Quality and Software Evolution

Software quality determines how well the software is designed (quality of design), and how well the software conforms to that design (quality of conformance).

In a software project, most of the cost is consumed in making changes rather than developing the software. Software evolution (maintenance) involves making changes in the software. Changes are required because of the following reasons:

1. Defects reported by the customer
2. New functionality requested by the customer
3. Improvement in the quality of the software
4. To adapt to new platforms

The typical evolution process is depicted in Figure 1.8. The figure shows that a change is requested by a stakeholder (anyone who is involved in the project) in the project. The second step requires analyzing the cost of implementing the change and the impact of the change on the related modules or components. It is the responsibility of an expert group known as the change control board (CCB) to determine whether the change must be implemented or not. On the basis of the outcome of the analysis, the CCB approves or disapproves a change. If the change is approved, then the developers implement the change. Finally, the change and the portions affected by the change are tested and a new version of the software is released. The process of continuously changing the software may decrease the quality of the software.

The main concerns during the evolution phase are maintaining the flexibility and quality of the software. Predicting defects, changes, efforts, and costs in the evolution phase

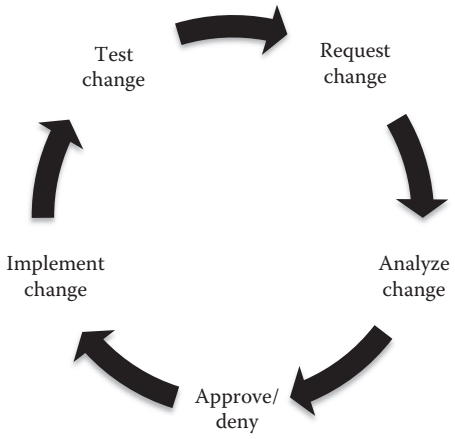


FIGURE 1.8
Software evolution cycle.

is an important area of software engineering research. An effective prediction can lead to decreasing the cost of maintenance by a large extent. This will also lead to high-quality software and hence increasing the modifiability aspect of the software. Change prediction concerns itself with predicting the portions of the software that are prone to changes and will thus add up to the maintenance costs of the software. Figure 1.9 shows the various research avenues in the area of software evolution.

After the detection of the change and nonchange portions in a software, the software developers can take various remedial actions that will reduce the probability of occurrence of changes in the later phases of software development and, consequently, the cost will also reduce exponentially. The remedial steps may involve redesigning or refactoring of modules so that fewer changes are encountered in the maintenance phase. For example, if high value of the coupling metric is the reason for change proneness of a given module. This implies that the given module in question is highly interdependent on other modules. Thus, the module should be redesigned to improve the quality and reduce its probability to be change prone. Similar design corrective actions or other measures can be easily taken once the software professional detects the change-prone portions in a software.

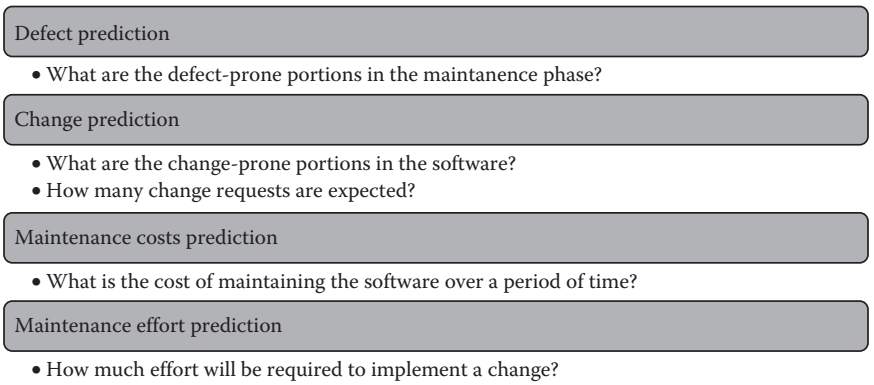


FIGURE 1.9
Prediction during evolution phase.

1.8.2 Software Quality Attributes

Software quality can be measured in terms of attributes. The attribute domains that are required to define for a given software are as follows:

1. Functionality
2. Usability
3. Testability
4. Reliability
5. Maintainability
6. Adaptability

The attribute domains can be further divided into attributes that are related to software quality and are given in Figure 1.10. The details of software quality attributes are given in Table 1.3.

1.8.3 Measures, Measurements, and Metrics

The terms measures, measurements, and metrics are often used interchangeably. However, we should understand the difference among these terms. Pressman (2005) explained this clearly as:

> A measure provides a quantitative indication of the extent, amount, dimension, capacity or size of some attributes of a product or process. Measurement is the act of determining a measure. The metric is a quantitative measure of the degree to which a product or process possesses a given attribute.

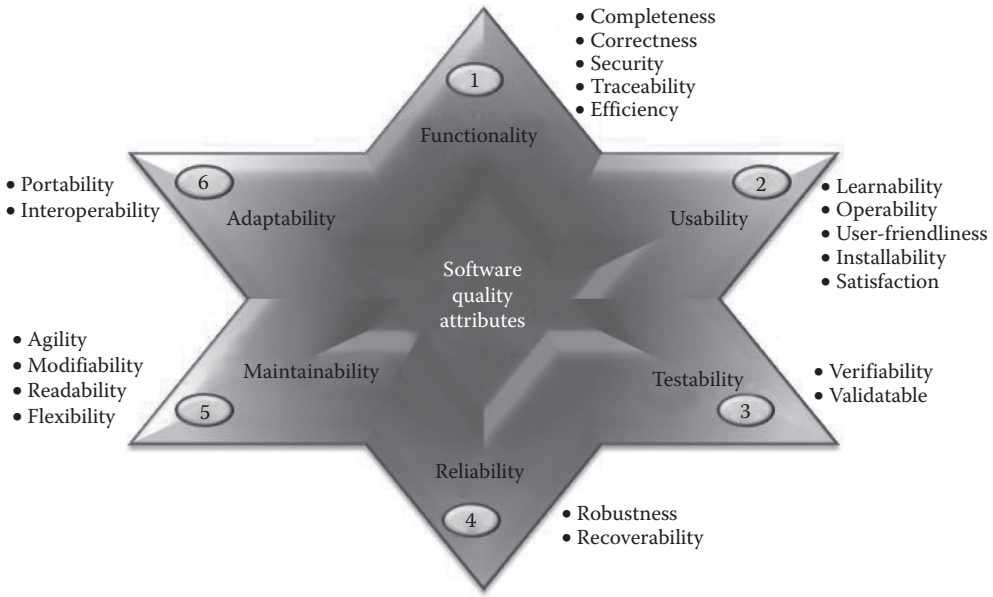


FIGURE 1.10
Software quality attributes.

TABLE 1.3

Software Quality Attributes

Functionality: The degree to which the purpose of the software is satisfied		
1	Completeness	The degree to which the software is complete
2	Correctness	The degree to which the software is correct
3	Security	The degree to which the software is able to prevent unauthorized access to the program data
4	Traceability	The degree to which requirement is traceable to software design and source code
5	Efficiency	The degree to which the software requires resources to perform a software function
Usability: The degree to which the software is easy to use		
1	Learnability	The degree to which the software is easy to learn
2	Operability	The degree to which the software is easy to operate
3	User-friendliness	The degree to which the interfaces of the software are easy to use and understand
4	Installability	The degree to which the software is easy to install
5	Satisfaction	The degree to which the user's feel satisfied with the software
Testability: The ease with which the software can be tested to demonstrate the faults		
1	Verifiability	The degree to which the software deliverable meets the specified standards, procedures, and process
2	Validatable	The ease with which the software can be executed to demonstrate whether the established testing criteria is met
Maintainability: The ease with which the faults can be located and fixed, quality of the software can be improved, or software can be modified in the maintenance phase		
1	Agility	The degree to which the software is quick to change or modify
2	Modifiability	The degree to which the software is easy to implement, modify, and test in the maintenance phase
3	Readability	The degree to which the software documents and programs are easy to understand so that the faults can be easily located and fixed in the maintenance phase
4	Flexibility	The ease with which changes can be made in the software in the maintenance phase
Adaptability: The degree to which the software is adaptable to different technologies and platforms		
1	Portability	The ease with which the software can be transferred from one platform to another platform
2	Interoperability	The degree to which the system is compatible with other systems
Reliability: The degree to which the software performs failure-free functions		
1	Robustness	The degree to which the software performs reasonably under unexpected circumstances
2	Recoverability	The speed with which the software recovers after the occurrence of a failure

Source: Y. Singh and R. Malhotra, *Object-Oriented Software Engineering*, PHI Learning, New Delhi, India, 2012.

For example, a measure is the number of failures experienced during testing. Measurement is the way of recording such failures. A software metric may be the average number of failures experienced per hour during testing.

Fenton and Pfleeger (1996) has defined measurement as:

> It is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules.

Software metrics can be defined as (Goodman 1993): "The continuous application of measurement based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products."

1.8.4 Descriptive, Correlational, and Cause–Effect Research

Descriptive research provides description of concepts. Correlational research provides relation between two variables. Cause–effect research is similar to experiment research in that the effect of one variable on another is found.

1.8.5 Classification and Prediction

Classification predicts categorical outcome variables (ordinal or nominal). The training data is used for model development, and the model can be used for predicting unknown categories of outcome variables. For example, consider a model to classify modules as faulty or not faulty on the basis of coupling and size of the modules. Figure 1.11 represents this example in the form of a decision tree. The tree shows that if the coupling of modules is <8 and the LOC is low then the module is not faulty.

In classification, the classification techniques take training data (comprising of the independent and the dependent variables) as input and generate rules or mathematical formulas that are used by validation data to verify the model predicted. The generated rules or mathematical formulas are used by future data to predict categories of the outcome variables. Figure 1.12 depicts the classification process. Prediction is similar to classification except that the outcome variable is continuous.

1.8.6 Quantitative and Qualitative Data

Quantitative data is numeric, whereas qualitative data is textual or pictorial. Quantitative data can either be discrete or continuous. Examples of quantitative data are LOC, number of faults, number of work hours, and so on. The information obtained by qualitative

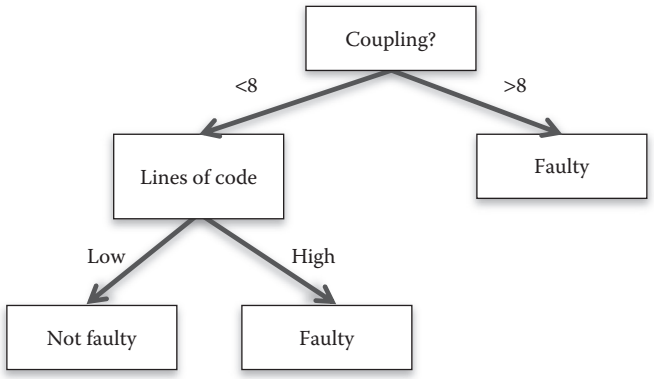


FIGURE 1.11
Example of classification process.