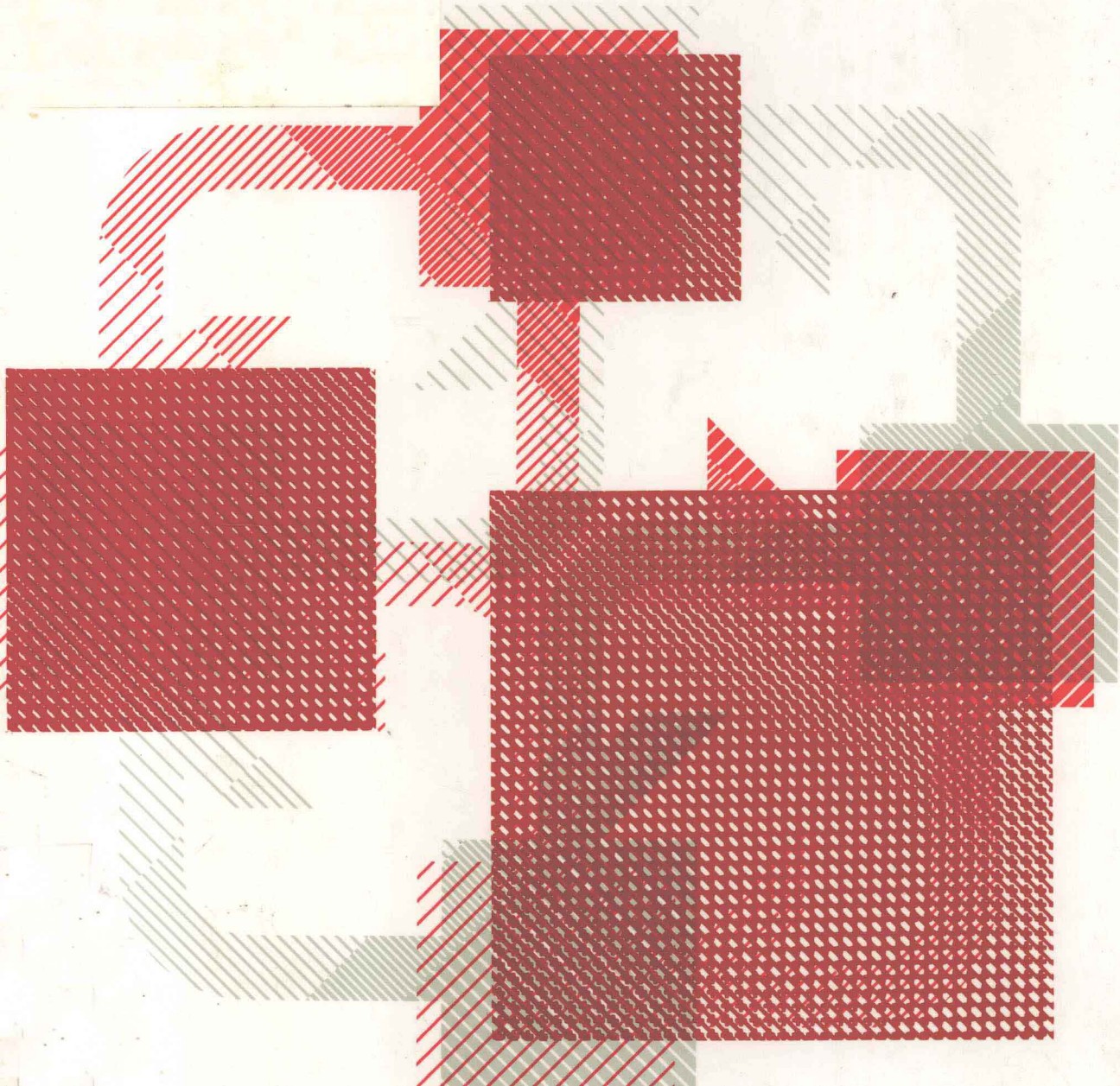
Stephen R. Schach

Software Engineering

Second Edition



Software Engineering

Second Edition

Stephen R. Schach
Vanderbilt University

IRWIN
Homewood, IL 60430
Boston, MA 02116

This symbol indicates that the paper in this book is made of recycled paper. Its fiber content exceeds the recommended minimum of 50% waste paper fibers as specified by the EPA.

Project Supervision: Science Typographers, Inc.
Cover and text designer: Harold Pattek
Compositor: Science Typographers, Inc.
Typeface: Times Roman
Printer: R. R. Donnelley & Sons Company

Printed in the United States of America
1 2 3 4 5 6 7 8 9 0 DOC 9 8 7 6 5 4 3

Library of Congress Cataloging-in-Publication Data

Schach, Stephen R.
Software engineering / Stephen R. Schach.—2nd ed.
p. cm.—(The Aksen Associates series in electrical and computer engineering)
Includes bibliographical references and indexes.
ISBN 0-256-12998-3
1. Software engineering. I. Title. II. Series.
QA76.758.S33 1993
005.1—dc20 92–31651

About the Author

Stephen R. Schach is Associate Professor of Computer Science at Vanderbilt University in Nashville, Tennessee. Dr. Schach received his M.Sc. at the Weizmann Institute of Science in Israel, and his Ph.D. at the University of Cape Town, South Africa. He has published over 70 technical papers in a variety of areas, including software engineering, software testing, and computer architecture. He is the author of the text *Practical Software Engineering* (Richard D. Irwin and Aksen Associates, 1992). Dr. Schach consults for industry and teaches courses in software engineering throughout the world. He is a member of the Association for Computing Machinery and the IEEE Computer Society.

The Aksen Associates Series in Electrical and Computer Engineering

Principles of Applied Optics
Partha P. Banerjee / University of Alabama, Huntsville
Ting-Chung Poon / Virginia Polytechnic and State University

Object-Oriented Engineering: Building Engineering Systems Using Smalltalk-80
John R. Bourne / Vanderbilt University

Discrete Event Systems: Modeling and Performance Analysis
Christos G. Cassandras / University of Massachusetts, Amherst

An Introduction to Fiber Optic Systems
John P. Powers / Naval Postgraduate School

Practical Software Engineering
Stephen R. Schach / Vanderbilt University

Software Engineering, Second Edition
Stephen R. Schach / Vanderbilt University

Introduction to Applied Statistical Signal Analysis
Richard Shiavi / Vanderbilt University

High Speed Communications Networks
Pravin Varaiya and Jean Walrand / University of California, Berkeley

Communications Networks: A First Course
Jean Walrand / University of California, Berkeley

Preface

"... a software product is a model of the real world, and the real world is constantly changing."

(*Software Engineering*, First Edition, 1990, page 9)

When I wrote those words explaining why change is an inherent part of software engineering, I did not realize that change is also an inherent part of software engineering textbooks, and for the same reason. In the three years since the first edition was published, there have been major changes in the software world. In particular, the object-oriented paradigm is no longer a curiosity for the elite few, but is now being used all over the world. Users are reporting that when products are designed in terms of objects, maintenance costs are reduced and ease of reuse of software components is promoted. As part of the move to object-oriented techniques, products are being implemented using object-oriented languages such as C++.

The U.S. Department of Defense language Ada is considerably less widely used than was anticipated in 1990. This is largely a consequence of the so-called Peace Dividend; less defense-related software is being developed. But it was also believed that Ada would be widely used in the nondefense sector.

This has not happened, probably because Ada is not an object-oriented language. Instead, C++ is the language of the future.

The software process is being closely examined, partly as a result of the efforts of the Software Engineering Institute (SEI) at Carnegie-Mellon University. More and more software organizations are using the capability maturity model (CMM) in order to improve their software process and thereby boost productivity. An integral part of this is the use of metrics to enable software developers to understand what is happening within their organizations. Thus software metrics are also becoming increasingly important.

A number of other topics that used to be of secondary importance are now major foci of software engineering interest. These include CASE tools (especially integrated CASE) and software reuse. The hope is that use of these technologies will increase productivity within those organizations that make use of them.

All these changes are reflected in the second edition of *Software Engineering*. In particular, the book is process-oriented, that is to say, a considerable effort has been made to arrange the material in a format that more closely reflects the process orientation of modern software engineering. The material is presented phase-wise. For example, Chapters 9 and 10 contain all the material on the design phase, including design methods, design testing, design reuse, metrics for the design phase, CASE tools for the design phase, and so on.

Notwithstanding the emphasis on modern developments and technology, the fundamental principles stressed in the first edition are stressed just as firmly in this edition. The first principle is analysis and comparison. A variety of techniques are described for each phase of the process; these techniques are then carefully contrasted. Because of the plethora of present-day software engineering techniques, it is important to select an appropriate one for the task at hand. The emphasis in this book on analysis and comparison alerts the reader to the need for careful choice and provides him or her with criteria for choosing wisely. The second principle is that the results of experiments in software engineering constitute a powerful tool for determining which techniques are to be preferred for a given situation. The third principle is that maintenance is a vital phase of the software process because, on average, maintenance consumes two-thirds of the total software budget. Maintenance must be planned for throughout the software development process. The fourth principle is that testing is not a phase that is performed just before the product is delivered to the client, nor is it an activity that is performed at the end of each phase of the software process. Instead, testing must be performed continually throughout the entire software development and maintenance process. The fifth principle is that all the documentation relating to a given phase of the software process must be completed before the next phase is started. In addition, the documenta-

tion must constantly be updated during maintenance, thus ensuring that the documentation always reflects the current version of the software.

The book is still essentially language-independent. The few code examples are in C++, rather than Ada. To be more precise, wherever possible the "C subset of C++" has been used. In addition, care has been taken to use as few C idioms as possible so that the material can also be understood by readers with little or no knowledge of C. The only place where C++ (rather than C) is employed is Chapter 9, and detailed explanations of specific C++ constructs have been provided there. Ada is used in one chapter near the end of the book entitled "Ada: A Case Study in Software Engineering." This material may be omitted if the instructor feels that it is inappropriate.

In this edition, additional material has been presented on those topics that have grown in importance since the first edition appeared, such as CASE tools, the spiral model, reuse, and metrics. New topics introduced include joint application design (JAD), the capability maturity model (CMM), and the formal specification language Z. At the same time, in order to keep the length of the book approximately the same as that of the first edition, a few topics (such as PAISLey) presented in the first edition have been omitted from this new edition. I feel that it is important that a textbook can be used in its entirety—presenting an instructor with twice as much material as he or she can conveniently cover in a course is generally counterproductive.

With regard to prerequisites, it is assumed that the reader is familiar with one high-level programming language such as Pascal, C, BASIC, COBOL, or FORTRAN. Although some of the examples are in C, no previous knowledge of C is needed. In addition, the reader is expected to have taken a course in data structures and to have acquired the familiarity with computers that is gained through carrying out programming exercises.

How the Second Edition Is Organized

This book is divided into three parts. The first part is introductory. The reader is introduced to software engineering and to the software process. An overview is given of tools such as stepwise refinement, cost–benefit analysis, and computer-aided software engineering (CASE) tools. Introductions to metrics and testing are also given.

The second part of this book is devoted to the various phases of the software process, from requirements through to maintenance. For each phase in turn, methods for performing that phase are described, as are testing techniques and CASE tools for that phase. Metrics and management techniques for each phase are also presented.

Software Engineering is designed to be used in a project-oriented course. The third part of this book accordingly consists of three chapters that can be presented while the class is completing the term project. Without these chapters, the instructor would have little or no material to teach during the final weeks while the last part of the project is being completed. The third part of *Software Engineering* consists of topics that do not directly relate to the software process, and can therefore be taught at the end of the course in parallel with the students finishing their project work.

As before, each chapter is concluded with a chapter review, suggestions for further reading in that area, a set of problems, and references for that chapter; the Bibliography for the book as a whole is presented at the end of the book. A new addition is Appendix B which contains a list of journals and conference proceedings that contain important articles on software engineering topics.

The Problem Sets

With regard to the problems included in this edition, each time I taught a course using the first edition, I gave the students a new problem set. For this edition I have selected what I consider to be the most interesting of the problems selected from the first edition and the new problem sets. I have also included my favorite among the various term projects I have set for my students.

As in the first edition, there are three types of problems. First, each chapter has a number of problems intended to highlight key points. These problems are self-contained; there is no need to visit a computer store to determine the specifications of various machines or to contact computer professionals to obtain information such as software productivity data. In general, computer salespeople and software professionals have neither the time nor the inclination to serve as sources of information for solving problems in textbooks, and students are not comfortable contacting busy individuals in order to obtain the data they need. For these reasons, the technical information for all of the problems can be found in this book.

Second, there is a software term project. Software engineering is a practical discipline, and purely theoretical courses are usually not very effective, so a practical project has been included in this book. Because so much software today is produced by teams rather than by individuals, the term project is designed to be solved by students working in teams of 3, the smallest number of team members that cannot confer over a standard telephone. The term project comprises 15 separate components, each tied in to the relevant chapter. For example, design methods is the major topic of Chapter 10, so in that chapter the component of the term project is concerned with designing the software for the

term project. By breaking up a large project into smaller, well-defined pieces, the instructor will be able to monitor the progress of the class more closely. The project is concerned with software for the Martha Stockton Greengage Foundation as described in Appendix A, but the structure of the project is such that the instructor may freely apply the 15 components to any other project of his or her choosing.

Because this book is designed for use by graduate students as well as upperclass undergraduates, the third type of problem is based on research papers in the software engineering literature. In each chapter an important paper has been chosen. The student is asked to read the paper and to answer a question relating to the contents of the paper. Of course, the instructor is free to assign any other research paper; to assist in this regard, the For Further Reading section at the end of each chapter includes a wide variety of relevant papers.

The Instructor's Manual contains detailed solutions to all the problems, as well as to the term project. The Instructor's Manual also contains transparency masters for all the figures in this book. The Instructor's Manual is available from Richard D. Irwin, Inc., Homewood, IL 60430.

Acknowledgments

I am indebted to those who reviewed this edition, including Kiumi Akingbehin, University of Michigan, Dearborn; Phil Bernhard, Clemson University; James Cardow, Air Force Institute of Technology; Betty Cheng, Michigan State University; Bob Goldberg, IBM; Ron New, Johns Hopkins University; Peter Jones, University of Western Australia; Everald E. Mills, Seattle University; and Fred Mowle, Purdue University.

It has been a real pleasure to work on a third book with Howard S. Aksen, publisher, editor, and friend. His unrivaled expertise has guided the project from start to finish.

Finally, I should like to thank my wife, Sharon, and our children, David and Lauren, for their understanding and forbearance whenever book deadlines clashed with family commitments. This book is dedicated to them, with love.

Stephen R. Schach

Contents