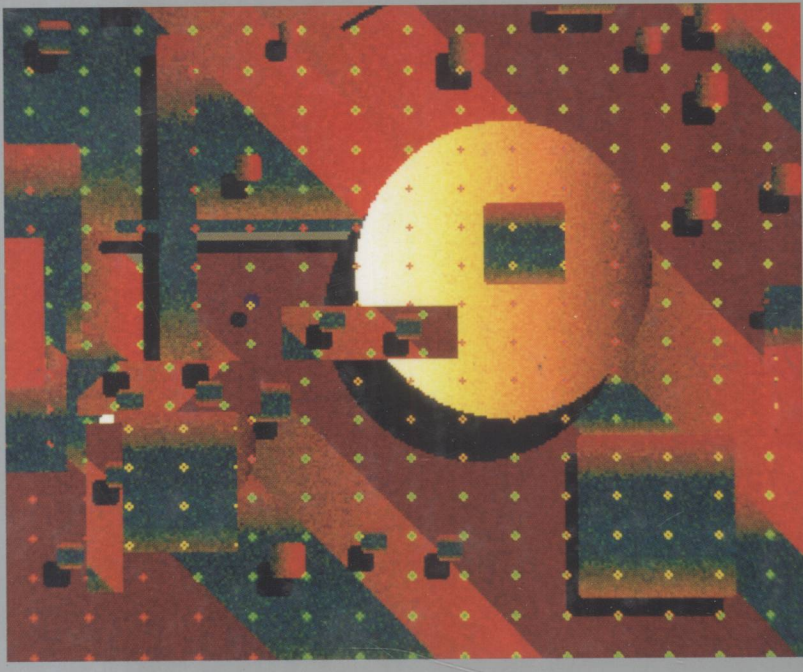
TEXTS IN COMPUTER SCIENCE

Software Reliability Methods



Doron A. Peled

Springer

Doron A. Peled

SOFTWARE RELIABILITY METHODS

Foreword by Edmund M. Clarke

With 50 Illustrations

Springer

Doron A. Peled
Computing Sciences
Bell Labs/Lucent Technologies
Murray Hill, NJ 07974, USA
doron@research.bell-labs.com

Library of Congress Cataloging-in-Publication Data
Peled, Doron, 1962–
Software reliability methods / Doron A. Peled.
p. cm. — (Texts in computer science)
Includes bibliographical references and index.
ISBN 0-387-95106-7 (alk. paper)
1. Computer software—Reliability. I. Title. II. Series.
QA76.76.R44 P317 2001
005—dc21 2001018395

Printed on acid-free paper.

Production managed by Allan Abrams; manufacturing supervised by Jerome Basma.
Photocomposed copy prepared from the author's LaTeX files.
Printed and bound by Maple-Vail Book Manufacturing Group, York, PA.
Printed in the United States of America.

9 8 7 6 5 4 3 2 1

ISBN 0-387-95106-7 SPIN 10774651

Springer-Verlag New York Berlin Heidelberg
A member of BertelsmannSpringer Science+Business Media GmbH

TEXTS IN COMPUTER SCIENCE

Editors
David Gries
Fred B. Schneider

Springer
New York
Berlin
Heidelberg
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

TEXTS IN COMPUTER SCIENCE

Alagar and Periyasamy, Specification of Software Systems

Apt and Olderog, Verification of Sequential and Concurrent Programs, Second Edition

Back and von Wright, Refinement Calculus

Beidler, Data Structures and Algorithms

Bergin, Data Structure Programming

Brooks, C Programming: The Essentials for Engineers and Scientists

Brooks, Problem Solving with Fortran 90

Dandamudi, Introduction to Assembly Language Programming

Fitting, First-Order Logic and Automated Theorem Proving, Second Edition

Grillmeyer, Exploring Computer Science with Scheme

Homer and Selman, Computability and Complexity Theory

Immerman, Descriptive Complexity

Jalote, An Integrated Approach to Software Engineering, Second Edition

Kizza, Ethical and Social Issues in the Information Age

Kozen, Automata and Computability

Li and Vitányi, An Introduction to Kolmogorov Complexity and Its Applications, Second Edition

(continued after index)

Foreword by Edmund M. Clarke

It is a great pleasure for me to write the foreword for Doron Peled's new book on software reliability methods. When I first opened the book, I was immediately impressed by the breadth of its coverage. It covers

- specification and modeling,
- deductive verification,
- model checking,
- process algebra,
- program testing, and
- state and message sequence charts.

In addition to describing the individual methods in considerable depth, it also discusses when each method is appropriate and the tradeoffs that are necessary in selecting among them. The different techniques are illustrated by many challenging exercises that can be used in conjunction with state of the art tools. It even tells where to access the tools on the web! I do not know of any other book that covers the same topics with such depth.

The book also describes the process of applying formal methods, starting with modeling and specification, then selecting an appropriate verification technique, and, finally, testing the resulting program. This knowledge is essential in practice, but is rarely covered in software engineering texts. Most books focus on a particular technique like program testing and do not cover other validation techniques or how several techniques can be used in combination. Because Doron has made significant contributions to the development of many of the validation techniques described in the book, his insights are particularly important on this critical issue.

The book is appropriate for a wide spectrum of people involved in the development of software. It is particularly appropriate for an upper level undergraduate level course on software reliability or a master's degree course in software engineering. In fact, it is sufficiently well annotated with pointers to other more advanced papers that it can be used as a reference source for software engineers engaged in code validation or by researchers in formal methods.

Having just completed a book on model checking with Doron, I am immensely impressed with both his talent as a computer scientist and his skill as a writer. I am sure that the present book will be an enormous success. I recommend it with great enthusiasm for anyone who is interested in the problem of software reliability.

Preface

Many books focus on increasing the quality of software through the use of formal methods. However, most books embrace one particular method, and present it as *the* suggested solution for the software reliability problem. This book presents a wider picture of formal methods, through a collection of notations and techniques. It compares them, and discusses their advantages and disadvantages.

One of the main challenges of formal methods is in transferring the technology developed by researchers to the software development community. Recently, we seem to be starting to have a better understanding of the important ingredients of formal methods tools. This manifests itself in the growing acceptance of such tools in the software and hardware development industry. Ideally, formal methods need to be intuitive to use (preferably using graphical interfaces), do not impose on the user an extensive learning period, and incur only small overhead to the development process. Formal methods are much more acceptable today than ten or twenty years ago, in particular in the hardware industry. Yet there is still a lively contention between different approaches.

The focus of this book is on describing the main principles of formal methods, through a collection of techniques. At the time of writing this book, there are already many advanced techniques that are not covered here. Techniques that deal with real-time and hybrid systems, advanced specification formalisms, and special data structures such as binary decision diagrams, were not included. The exclusion of specific material does not mean that the methods presented here are superior to the ones omitted. Nevertheless, the algorithms and methods described here are being used in state-of-the-art software reliability tools. The selection is merely intended to present the subject of formal methods in a way that seems deductive. However, it is impossible to refrain from preferring to include subjects that are closer to one's own research The main themes used throughout this book are *logic* and *automata theory*. The interested reader can find details of advanced approaches in other books and research papers listed at the end of relevant chapters.

Studying formal methods is incomplete without hands-on experience with some tools. This book includes various exercises and projects, which may be performed using software reliability tools. There are several running examples

that are used in different chapters. An effective way to learn formal methods and their strengths and weaknesses is to follow these examples throughout the different chapters in which they occur. In some cases, a later chapter further elaborates on a running example that was presented as an exercise in a previous chapter. This also serves the purpose of helping readers check their solutions to previous exercises (instead of providing an explicit solution). The readers are encouraged to check if some of the additional intuition gained about the running example may help in improving their solutions to previous exercises.

Most exercises and projects presented here can be performed using a choice of tools. While some of the software reliability tools are subject to nontrivial license fees, many of them can be used free of charge for nonprofit purposes. This usually involves downloading the tool from its worldwideweb page and installing it according to the instructions provided there. At the end of relevant chapters, some tools and their corresponding web pages are listed. Notice that even tools that can be used without acquiring a license often require an agreement letter to be sent to the developers of the tool, committing to their terms of use. In many cases, such terms restrict the use of the tool for academic purposes only, and maintain no responsibility for damage that may be incurred by using it. Since web pages and web addresses tend to change, and since new tools are constantly being constructed, replacing existing ones, one cannot guarantee that the provided web information will remain up to date for long. Moreover, it is not guaranteed that the tools will work under any particular environment.

Different communities have different interests in formal methods. It is of course impossible to present a book that will appeal equally to managers, software developers, quality assurance teams and researchers alike. Nevertheless, I tried to include material that would be interesting to members of each one of these groups. Consequently, the reader may want to skip sections that may seem too theoretical, or too technical. It should be pointed out that the focus of this book is mainly on *techniques* rather than on *methodology*.

Some of the formal methods presented in this book are described together with the corresponding algorithm. Understanding the algorithms is usually not crucial for using the methods, but may give a deeper perspective on how they work. Most of the mathematical proofs involving the methods described were omitted. In some cases, proof sketches are included, to add more intuition.

The author would like to thank the following people for enlightening discussions and comments related to this book: Nina Amla, Christel Baier, David Basin, Shai Ben-David, Roderick Bloem, Glenn Bruns, Ed Clarke, Dennis Dams, Xiaoqun Du, Kousha Etessami, Amy Felty, Elsa Gunter, Doug Howe, Orna Kupferman, Bart Knaack, Bob Kurshan, Bengt Jonsson, Leonid Libkin, Anca Muscholl, Kedar Namjoshi, Wojciech Penczek, Kavita Ravi, Natarajan Shankar, Natasha Sharygina, Marian Srenby, Richard Tefler, Wolfgang

Thomas, Moshe Vardi, Igor Walukiewicz, Thomas Wilke, Mihalis Yannakakis and Lenore Zuck. Indeed, one of the great benefits of writing such a book is the opportunity to further learn from the suggestions and comments of practitioners and experts of the particular subject.

Quoting from Lewis Carroll's adventure books is hardly original. However, it is little known that Charles Lutwidge Dodgson, who wrote under the pen name Lewis Carroll, was a mathematician interested in the visual representation of logic. His 'biliteral' and 'triliteral' diagrams are predecessors of Karnaugh maps, representing logic in a way that can be easily taught and understood, a recent trend in many formal methods.

Doron Peled, March 2001, Murray Hill, NJ

Contents

List of Figures

1. Introduction

'Where shall I begin, please your Majesty?' he asked. 'Begin at the beginning,' the King said, very gravely, 'and go on till you come to the end; then stop.'

Lewis Carroll, *Alice's Adventures in Wonderland*

During late 1999, the world waited, with growing concern, for the change of the calendar into the year 2000. The focus was on some potential damage from online computers that control vital systems. This damage could occur because of the mere change of the calendar year and the way years of the twentieth century were traditionally represented in computer memory, using only the least two significant digits from 00 to 99. This surprisingly small detail made some people expect extreme damage. It could have affected electronic systems driven by software, such as verifying traffic control, atomic missiles, nuclear reactors, banking systems, pension plans, electricity and water supply. The US alone spent over 100 billion dollars on combating this, so called, 'Y2K-bug.' Just prior to that date change, some people had escaped into self made shelters, while flashlights and bottled water were a popular demand. Joint teams of the US and Russian military spent the night of December 31 1999 at the North American Aerospace Defense Command (NORAD). Together they monitored the world's skies, as a precaution against a possible computer error that could cause an unruly launch of missiles. Midnight, December 31 1999 has passed into the new millenium with no significant events, except for a few minor glitches.

Computer systems control many aspects of our lives. Telephone systems, store checkout registers, ticket reservation systems, medical systems, financial systems, are all highly computerized. Data communication between computers replaces, in most cases, the use of actual paper money transfer. Computers are even responsible for many of the activities required for flying commercial airplanes. Failure of computerized systems have already caused grave consequences, including fatal accidents, shutting down of vital systems, and loss of money.

The software development industry has grown over the last few decades at an unprecedented pace. Hardware, and in particular memory costs, kept decreasing. The internet has practically transformed the world into a big