SOFTWARE MAINTENANCE

Concepts and Practice

SECOND EDITION

Penny Grubb (University of Hull, UK) &
Armstrong A Takang (Software Systems Consultant, USA)

World Scientific
New Jersey • London • Singapore • Hong Kong

Acknowledgements

To our families, especially

- George and Danny (and thanks for all that proof-reading)
- Ayem, Bessem and Nyente.

The authors would like to thank colleagues and friends in the various health care facilities across the world whose software trials and tribulations have been amalgamated into the Acme Health Clinic case studies used in the book.

Thanks also to Steven Patt and his colleagues for editorial assistance.

Preface

Aims and Objectives

The purpose of this book is to explore the key issues underpinning software change and to discuss how these issues impact on the implementation of changes to software systems. The motivation for the book came from the need for texts dealing directly with challenges that software engineers face when modifying complex software systems. The extent of this challenge can be seen in the cost of modifying software. This cost can reach 70% of the total life-cycle cost [4, 36, 176]. Software maintenance is recognised as a key area in software engineering [9, 163]. Despite this, many mainstream software engineering courses are biased towards the development of new software systems at the expense of issues surrounding changes to these systems after they become operational [70].

Our intention is to produce a text that presents:

- a coherent and comprehensive coverage of software change concepts;
- a theoretical base for the skills required to effect, control and manage changes to evolving software systems;
- a framework for understanding and applying current maintenance techniques and methods to solve problems.

This is not a cookbook; there is no set of cut and dried rules for dealing with the problems of software maintenance. An elegant and workable solution in one situation may be completely inadequate for the same problem in a different environment. Nonetheless, it is essential for software engineers to have a sound understanding of software maintenance for several reasons. Firstly, it is common wisdom that a large part of finding a solution to a problem lies in understanding it. Secondly, an insight into the issues underpinning software maintenance can help in the formulation of an adequate framework that can be used to guide the development of appropriate support tools. This framework also enables researchers to identify potential research questions and compare research findings.

Target Audience

This book is aimed at students, academics and professionals who have an interest in the development and maintenance of software systems.

It is intended as a reference text and also as a course book for software maintenance, software evolution and general courses on advanced software engineering. It can also serve as an introductory text for those intending to engage in research into software maintenance.

For undergraduate study, the book aims to raise awareness of software maintenance issues, for example the need to develop programs that cater for the evolutionary tendency of software systems. This not only provides a grounding in the discipline, but is also a preparation for life in the commercial world. The first job of many graduates going into the software industry involves the maintenance of existing systems rather than the development of new systems [187, 282]. Additionally, the book is intended to complement other undergraduate software engineering and programming courses.

For software professionals, the text provides a collection of definitions for some of the commonly used terms. This is important because of the plethora of terms and jargon in use [211]. In addition, the case studies and real world examples provided should help during in-service training or refresher courses on software maintenance.

Structure and Organisation of this Book

The book is organised into five parts.

The first part looks at the context of software maintenance. It introduces the basic concepts and the framework within which maintenance operates. Underlying theory is introduced by looking at the fundamentals of software change, but real world considerations are also introduced at this stage. This part of the book concludes with a look at how software development and maintenance life-cycles are modelled.

The second part of the book goes through the activities that take place during maintenance, starting with understanding the system to be changed, through the specifics of making the change and testing the modified system, to the managerial issues and decision-making that accompanies the process.

The third part looks at means of measurement and assessment, both of the overall process and of the components of software and software maintenance, showing how to keep track and provide objective assessment.

These first three parts of the book look at what software maintenance is and how to do it. In total they build the case for maintainability in systems.

The fourth part looks at how these lessons can be used in the building of better systems.

The fifth and final part looks at research areas and the future for the discipline of software maintenance.

Each major section is preceded by a number of discussion points aimed at provoking thought about some of the fundamental issues.

Exercises throughout the book vary from straightforward questions on the details of the text, to more complex role-playing projects where the reader is asked to put themselves into a particular maintenance context and think through a specific problem.

Both minor and major case studies are used throughout to relate the material to what is happening at the coal face of software maintenance.

Contents