Undergraduate Topics in Computer Science (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems. Many include fully worked solutions.

More information about this series at http://www.springer.com/series/7592

Neil Walkinshaw

# Software Quality Assurance

## Consistency in the Face of Complexity and Change

Springer

Neil Walkinshaw
Department of Computer Science
University of Leicester
Leicester
UK

ISSN 1863-7310 ISSN 2197-1781 (electronic)
Undergraduate Topics in Computer Science
ISBN 978-3-319-64821-7 ISBN 978-3-319-64822-4 (eBook)
DOI 10.1007/978-3-319-64822-4

Library of Congress Control Number: 2017947829



Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

*To Emma, Iona, and Dougie.*

# Preface

> "Let's think the unthinkable, let's do the undoable. Let us prepare to grapple with the ineffable itself, and see if we may not eff it after all."
>
> Douglas Adams, *Dirk Gently's Holistic Detective Agency*

This book is an introduction for students to the main principles and some of the most popular techniques that constitute 'software quality assurance'. It is worth emphasising from the outset that this book is *not* a reference book. There are already plenty of excellent comprehensive Software Engineering reference books in print.

Instead, this book seeks to provide a focus on Quality Assurance that typical, more generic Software Engineering reference books do not. The goal is to to do so in such a way that the book can be read from cover to cover throughout the course of a typical university module. Specifically, this book aims to be:

- **Concise:** It aims to be small enough to be readable in its entirety over the course of a typical software engineering module.
- **Explanatory:** When topics are covered, it is important not merely to describe *what* they are, but also *why* they are the way they are – describing what events, technologies, and individuals or organisations helped to shape them into what they are now.
- **Applied:** Topics will be covered with a view to giving the reader a good idea of how they can be applied in practice, and by pointing where possible to evidence about their efficacy.

Quality Assurance is often presented and discussed in somewhat utilitarian terms, as a set of necessary, occasionally tedious, techniques; required reading for anybody who aspires to become a capable, reliable Software Engineer. This brings us to the final, slightly more nebulous objective of this book: To convince the reader that there is much, much more to Quality Assurance than that.

We inhabit a world in which software is increasingly pervasive – controlling everything from light bulbs in homes to smart phones, cars, planes, power stations, and

voting machines. Failures in software quality can have and have had disastrous consequences. There is an urgent need for a widespread appreciation of how precarious software quality can be, and how it can improved and ensured.

Although the application of Quality Assurance techniques can become 'tedious', this misses what are (for the author at least) the real attractions. The subject is not only necessary, but academically fascinating too. There is no way of *guaranteeing* that a software system will 'succeed' - that it will not contain bugs, satisfy the customer, and be delivered on time and at cost. The task of building complex systems according to complex, continuously changing requirements, in a limited amount of time, within a limited budget, whilst managing large teams of developers, is enormously challenging. There is no single 'best' solution, and there are so many open (often surprising) problems.

## *Acknowledgements*

This book is an extension of the course notes for the "Software Quality Assurance and Metrics" course at the University of Leicester, jointly taught to under- and postgraduate students. The course was originally taught by Helge Janicke (now at De Montfort University) until I took over as convenor in 2013. Although the course has changed in several respects, I am very grateful to Helge for developing the initial structure, and thus setting the direction for a large portion of the subject-matter covered in this book.

Throughout the writing of the book, several undergraduate and postgraduate students have been kind enough to provide valuable feedback on its contents. Many thanks especially to Cara Bateman, Anita Lad, Shriya Malhotra, and Sylvester Saracevas.

Finally, I owe a debt of gratitude to the editorial team at Springer, and Ralf Gerstner in particular, for their valuable support, feedback, and patience.

# Contents