

Validation of Evolving Software

Hana Chockler · Daniel Kroening
Leonardo Mariani · Natasha Sharygina
Editors

Validation of Evolving Software

Editors

Hana Chockler
Department of Informatics
King's College
London
UK

Leonardo Mariani
Department of Informatics, Systems
and Communication
University of Milano Bicocca
Milano
Italy

Daniel Kroening
Department of Computer Science
University of Oxford
Oxford
UK

Natasha Sharygina
Formal Verification and Security Lab,
Informatics Department
Università della Svizzera Italiana
(University of Lugano)
Lugano
Switzerland

ISBN 978-3-319-10622-9 ISBN 978-3-319-10623-6 (eBook)
DOI 10.1007/978-3-319-10623-6

Library of Congress Control Number: 2015942177

Springer Cham Heidelberg New York Dordrecht London
© Springer International Publishing Switzerland 2015

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media
(www.springer.com)

Preface

In our everyday life, we rely on the availability and flawless functioning of complex distributed infrastructures, such as electricity, water, communication, transportation and environmental management. This infrastructure is based on large computerized systems for monitoring and control. Technological innovation offers opportunities for more efficient infrastructure, but innovation in infrastructures and the resulting improvement in quality of life is hindered by the danger of changes and upgrades in existing systems. Indeed, a change can introduce errors resulting in crashes, loss of existing functionality or incompatibility between versions, which can result in major service outages. To make matters worse, most of these systems are networked systems, in which the upgrades are naturally done gradually, so several versions have to co-exist in the same system.

Currently, all practices of error detection and validation rely on re-validating the whole system, which is very time-consuming and expensive; fault localization is mainly manual and driven by experts' knowledge of the system; fault fixing often introduces new faults that are hard to detect and remove. The cost of validation, therefore, dominates the maintenance costs of the software (it has been estimated that the cost of change control can be between 40 % and 70 % of the life cycle costs [GT05]). As a consequence, project managers are often reluctant to authorize new features or even bug fixes. Citing one project manager, "Upgrading a large and complex embedded system is akin to upgrading the software of a car while the car's engine is running, and the car is moving on a highway. Unfortunately, we don't have the option of shutting the whole system down while we upgrade and verify a part of it." Infrastructure upgrades are done only once the existing infrastructure performs below acceptable levels, and a new version of software is fully re-verified and re-certified, which is clearly a very lengthy and expensive process. The situation is only getting worse because of shorter product lifecycles and the increasing complexity and scale of software systems, making the problem of efficient validation and certification of changes especially acute.

In this book, we describe common errors resulting from introducing changes and upgrades in an existing software system and suggest a mix of methodology and technologies in order to perform efficient validation of changes in complex systems.

We propose to start the validation process as early as possible and to apply a mix of static and dynamic analysis techniques for reliable validation. We introduce a novel validation technology, based on a tight integration between static and dynamic components—a *hybrid* technology—and show that it can perform efficient and scalable validation of changes and upgrades even in very large and complex software.¹

¹This book presents the technology and methodology developed as a part of the PINCETTE project, in the framework of the FP7 Program of the European Community under the call FP7-ICT-2009-5.

Contents

Part I Introduction

| | | |
|----------|---|-----------|
| 1 | Introduction | 3 |
| | Hana Chockler, Daniel Kroening, Leonardo Mariani and Natasha Sharygina | |
| 2 | Challenges of Existing Technology | 7 |
| | Hana Chockler, Daniel Kroening, Leonardo Mariani and Natasha Sharygina | |
| 3 | Complementarities Among the Technologies Presented in the Book | 19 |
| | Hana Chockler, Daniel Kroening, Leonardo Mariani and Natasha Sharygina | |

Part II Static Analysis

| | | |
|----------|--|-----------|
| 4 | Lightweight Static Analysis Check of Upgrades in C/C++ Software | 25 |
| | Hana Chockler and Sitvanit Ruah | |
| 5 | Function Summarization-Based Bounded Model Checking | 37 |
| | Ondrej Sery, Grigory Fedyukovich and Natasha Sharygina | |
| 6 | Incremental Upgrade Checking | 55 |
| | Ondrej Sery, Grigory Fedyukovich and Natasha Sharygina | |
| 7 | Optimizing Function Summaries Through Interpolation | 73 |
| | Simone Fulvio Rollini, Leonardo Alt, Grigory Fedyukovich, Antti Eero Johannes Hyvärinen and Natasha Sharygina | |

Part III Dynamic Analysis

- 8 RADAR: Dynamic Analysis of Upgrades in C/C++ Software** 85
 Fabrizio Pastore, Leonardo Mariani, Alberto Goffi,
 Manuel Oriol and Michael Wahler
- 9 G-RankTest: Dynamic Analysis and Testing of Upgrades
 in LabVIEW Software** 107
 Leonardo Mariani, Oliviero Riganelli, Mauro Santoro
 and Ali Muhammad

Part IV Common Preprocessing and Hybrid Analysis

- 10 Measuring Change Impact on Program Behaviour.** 125
 Ajitha Rajan and Daniel Kroening
- 11 Static/Dynamic Test Case Generation For Software Upgrades
 via ARC-B and Deltatest.** 147
 Pietro Braione, Giovanni Denaro, Oliviero Riganelli,
 Mauro Baluda and Ali Muhammad
- 12 Regression Checking of Changes in C Software** 185
 Fabrizio Pastore, Leonardo Mariani, Antti Eero Johannes Hyvärinen,
 Grigory Fedyukovich, Natasha Sharygina, Stephan Sehestedt
 and Ali Muhammad
- Bibliography** 209