Model-Based Software Performance Analysis

Vittorio Cortellessa • Antinisca Di Marco •
Paola Inverardi

Model-Based Software Performance Analysis

Springer

Vittorio Cortellessa
Antinisca Di Marco
Paola Inverardi
Dipartimento di Informatica
Università di L'Aquila
Via Vetoio 1
Coppito, 67010 L'Aquila
Italy
vittorio.cortellessa@univaq.it
antinisca.dimarco@univaq.it
paola.inverardi@univaq.it

ISBN 978-3-642-13620-7 e-ISBN 978-3-642-13621-4
DOI 10.1007/978-3-642-13621-4
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011927935

ACM Computing Classification (1998): D.2, C.4, D.4.8, I.6

© Springer-Verlag Berlin Heidelberg 2011
This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.
The use of general descriptive names, registered names, trademarks, 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.

Cover design: KünkelLopka GmbH

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

This book is dedicated to the town of L'Aquila, destroyed by an earthquake on April 6th, 2009. L'Aquila is the locus of our past births in life, in work, in love. As for the future we wish to L'Aquila its own re-birth to host again many lives, many works, many loves.

Vittorio, Antinisca and Paola

Preface

Goal of the Book

In the last decade there has been a growing interest in the research and software industry communities toward techniques, methods and tools that allow one to manage system performance concerns in the software developer domain. Poor performance can often be the cause of software project failure, and the need to address performance concerns during software development is fully acknowledged. One of the main impediments to progress in this field lies in the different cultures in software and performance and in the lack of standard practices and tool support. One promising direction to bridge this gap is the one described in this book. Model-based Software Performance Analysis is the research domain that introduces performance concerns into the scope of software models, thus allowing the developer to carry out performance analysis all along the software lifecycle.

The goal of the book is to provide the cross-knowledge that allows developers to face software performance issues since the very early phases of the software development. On one hand, we provide the basic concepts of performance analysis. On the other hand, we introduce the readers to the problem of making a performance analysis a common practice in the software development process and describe the most representative methodologies proposed to annotate and translate software models into performance models.

For the sake of book uniformity, we do not report here applications of approaches to case studies from the real world (except for a few cases), because the intent of the book is to provide an introduction to this domain. We leave to possible future editions the treatment of real case studies and the lessons learned from them.

A Bird's-Eye View of Chapters Contents

Chapter 1 embeds the software performance analysis activity in the wider framework of non-functional validation. It explains how different non-functional attributes contribute to the quality of a software product. Thereafter we clarify the

difference between software performance analysis (where static/dynamic software models are explicitly built and play a crucial role in the analysis process) and system performance analysis (where the software is only meant as a system synthetic workload).

Chapter 2 gives an overview of the most widely adopted software notations, including the techniques for extending UML. Note that this chapter does not aim at presenting notational details, as most of the readers should be familiar with the majority of this chapter concepts.

Chapter 3 gives an overview of the most widely adopted performance notations. The notations that will be used in the following chapters are treated to a larger extent. The others are only briefly surveyed and mentioned for completeness purposes.

Chapter 4 focuses on the software performance process. It introduces the software lifecycle phases outlining the distance between software artifacts and performance models.

Chapter 5 introduces the reader to the transformation-based approaches that support the generation of performance models from software models. In particular, it describes how software models can be annotated with performance parameters. Then an overview of the major supporting tools for model transformations in this domain is presented. The chapter ends with a classification and taxonomy of the existing approaches.

Chapter 6 provides techniques and tools to solve the performance models obtained through model transformations. Analytical, approximate and simulation-based techniques are described.

In Chap. 7 the major trends that may affect the future evolution of software performance analysis are presented. They address very different areas as they span from the relation of performance and Model Driven Architecture to the need of a software performance ontology.

In terms of novelty of contents: Chaps. 1–3 and Chap. 6 contain well assessed notions in the field; Chap. 4 introduces a new view in the integration of performance into a software lifecycle; Chap. 5 describes techniques and tools that have influenced the discipline in the last decade; finally, Chap. 7 provides a glimpse to the current research trends, thus sketching the possible evolutions of the discipline in the next decade.

Figure 0.1 summarizes this bird's-eye view of the chapters and their relationships.

Using the Book as Teaching Text

The book is not primarily intended as a teaching text. However, it can be used to complement more traditional Software Engineering teaching texts to provide a deeper insight of model software performance analysis. To this extent it provides chapters on the basic software and performance modeling notions and describes in detail some transformational techniques. Instructors may choose their favorite transformational approach and follow the detailed case study presentation as a guideline to present the major concepts underlying this discipline.

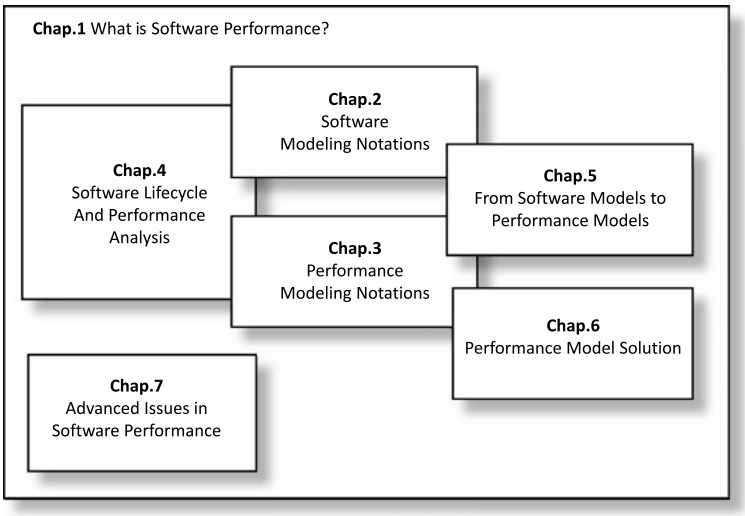

Fig. 0.1 Book chapters organization

Acknowledgments

Quite a large number of people have contributed over the years to this book contents, so it is difficult to mention all of them in a few lines.

First, thanks go to the "mother" and "father" of the software performance discipline, namely Connie Smith and Murray Woodside. Without their pioneering work, the whole community would likely not exist today.

Our earlier co-authors in this discipline have influenced, in different ways, the content of this book and have contributed to our passion in this field. Many thanks to Simonetta Balsamo, Giuseppe Iazeolla, Raffaela Mirandola and Vincenzo Grassi.

Special thanks go to Jane Hillston, who has allowed us to use her excellent lecture notes, and to Ed Lazowska and his co-authors, whose book is still today a reference point for us and our students. Also thanks to Dorina Petriu, for her extensive work in the area of Software Performance Engineering. Part of their work has been reported in this book.

Thanks are due to our Ph.D. students Pasqualina Potena, Catia Trubiani and Luca Berardinelli, who, besides sustaining us with their enthusiasm and criticism, have also concretely helped us with figures and tables. And thanks go to all our former, current and future students that, with their daily work, represent the lymphatic system of our working life, and contribute to our human and scientific growth.

Last but not least, we would like to thank the reviewers of this book, Andrea Polini and Jose Merseguer, who have helped us to largely improve the book's organization and contents. Many thanks for your deep and very useful reviews! Special thanks go to Ralf Gerstner, our editor at Springer, for his great patience and dedication.

Finally we would like to thank our families, because from the very beginning they have sustained our dreams (all of a different nature) and have allowed such dreams to become true. Every day it is clear to us that the existence of our families is per se an immense value that cannot be replaced by any other thing.

L'Aquila, Italy

Vittorio Cortellessa
Antinisca Di Marco
Paola Inverardi

Contents

1 What Is Software Performance? . . . 1
1.1 Non-functional Validation of Software Systems . . . 2
1.2 Performance as a Non-functional Attribute . . . 4
1.3 System vs. Software Performance Analysis . . . 6

2 Software Modeling Notations . . . 9
2.1 Basic Notations . . . 9
2.1.1 Automata . . . 11
2.1.2 Process Algebras . . . 12
2.1.3 Petri Nets . . . 14
2.1.4 Message Sequence Charts . . . 15
2.2 Unified Modeling Language . . . 17
2.2.1 E-commerce System . . . 18
2.2.2 Use Case Diagram . . . 19
2.2.3 Component Diagram . . . 20
2.2.4 Interaction Diagram . . . 22
2.2.5 Activity Diagram . . . 25
2.2.6 State Machine Diagram . . . 30
2.2.7 Deployment Diagram . . . 32
2.2.8 Profiling UML . . . 33

3 Performance Modeling Notations . . . 35
3.1 Markov Processes . . . 36
3.2 Queueing Networks . . . 39
3.2.1 QN Definition . . . 39
3.2.2 QN Parameterization . . . 40
3.3 Execution Graphs . . . 42
3.4 Layered Queueing Networks . . . 46
3.5 Stochastic Petri Nets . . . 49
3.6 Stochastic Process Algebras . . . 52
3.7 Simulation Models . . . 54

3.8 UML Profile for Schedulability, Performance and Time 55
3.8.1 PAprofile: Stereotypes and Tagged Values 56

4 Software Lifecycle and Performance Analysis 65
4.1 Software Lifecycle 65
4.2 Performance Analysis Within the Lifecycle 67
4.3 A Simple Application Example 74

5 From Software Models to Performance Models 79
5.1 A General Framework for Model Transformation 79
5.2 Some Transformational Approaches at Work 80
5.2.1 UML-ψ: From UML to a Simulation Model 81
5.2.2 From UML to a Layered Queueing Network 92
5.2.3 SAP•one: From UML to a Queueing Network 115
5.3 Other Transformational Approaches 131
5.3.1 Queueing Network Based Methodologies 131
5.3.2 Petri Net-Based Approaches 133
5.3.3 Methodologies Based on Simulation Methods 134
5.4 Discussion of the Approaches 135
5.5 Desirable Attributes of Software Performance Analysis Techniques 139

6 Performance Model Solution 141
6.1 Model Solution: Foundations and Techniques 142
6.1.1 Operational Analysis 143
6.1.2 Solution Techniques and Related Notations 148
6.1.3 Simulation 153
6.2 Model Solution: Tools 154
6.2.1 SHARPE—Multiple Performance Model Notations 154
6.2.2 SPE·ED—Execution Graphs and Queueing Networks . . . 155
6.2.3 GreatSPN—Stochastic Petri Nets 156
6.2.4 TimeNET—Stochastic Petri Nets 157
6.2.5 TwoTowers—Stochastic Process Algebras 158

7 Advanced Issues in Software Performance 159
7.1 Software Performance and Model-Driven Architecture 159
7.2 Interpretation of Performance Analysis Results 164
7.3 Performance-Based Software Reconfiguration 165
7.3.1 Allowed Reconfigurations 169
7.3.2 Issues to Address 169
7.4 A Unifying Ontology for Software Performance 171
7.4.1 Three Meta-models for Software Performance 172
7.4.2 Building an Ontology from Common Entities: A Bottom–Up Approach 177
7.4.3 Expressiveness Issues: A Top–Down Process 180

References 183

Index 189