Verification and Validation in Systems Engineering

Mourad Debbabi · Fawzi Hassaïne · Yosr Jarraya ·
Andrei Soeanu · Luay Alawneh

Verification and Validation in Systems Engineering

Assessing UML/SysML Design Models

Springer

Dr. Mourad Debbabi
Computer Security Laboratory
Concordia Institute for Information
Systems Engineering (CIISE)
Faculty of Engineering and Computer Science
Concordia University
1455, Boulevard de Maisonneuve West, EV 7-642
Montreal, Quebec, H3G 1M8 Canada
debbabi@ciise.concordia.ca

Dr. Yosr Jarraya
Computer Security Laboratory
Concordia Institute for Information
Systems Engineering (CIISE)
Faculty of Engineering and Computer Science
Concordia University
1455, Boulevard de Maisonneuve West, EV 7-642
Montreal, Quebec, H3G 1M8 Canada
y_jarray@encs.concordia.ca

Luay Alawneh
Department of Electrical and
Computer Engineering
Faculty of Engineering and Computer Science
Concordia University
1455, Boulevard de Maisonneuve West,
EV 13-173
Montreal, Quebec, H3G 1M8 Canada
l_alawne@encs.concordia.ca

Dr. Fawzi Hassaïne
Capabilities for Asymmetric and
Radiological Defence and Simulation
Defence Research and Development
Canada – Ottawa
3701 Carling Avenue
Ottawa, ON, Canada K1A 0Z4
fawzi.hassaine@drdc-rddc.gc.ca

Andrei Soeanu
Computer Security Laboratory
Concordia Institute for Information
Systems Engineering (CIISE)
Faculty of Engineering and
Computer Science
Concordia University
1455, Boulevard de Maisonneuve West,
EV 7-642
Montreal, Quebec, H3G 1M8 Canada
a_soeanu@ciise.concordia.ca

ISBN 978-3-642-15227-6 e-ISBN 978-3-642-15228-3
DOI 10.1007/978-3-642-15228-3
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2010936711

ACM Codes: D.2, C.4, K.6

© Springer-Verlag Berlin Heidelberg 2010
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: KuenkelLopka GmbH, Heidelberg

Printed on acid-free paper

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

To our families and friends!

Preface

At the dawn of the 21st century and the information age, communication and computing power are becoming ever increasingly available, virtually pervading almost every aspect of modern socio-economical interactions. Consequently, the potential for realizing a significantly greater number of technology-mediated activities has emerged. Indeed, many of our modern activity fields are heavily dependant upon various underlying systems and software-intensive platforms. Such technologies are commonly used in everyday activities such as commuting, traffic control and management, mobile computing, navigation, mobile communication. Thus, the correct function of the forenamed computing systems becomes a major concern. This is all the more important since, in spite of the numerous updates, patches and firmware revisions being constantly issued, newly discovered logical bugs in a wide range of modern software platforms (e.g., operating systems) and software-intensive systems (e.g., embedded systems) are just as frequently being reported.

In addition, many of today's products and services are presently being deployed in a highly competitive environment wherein a product or service is succeeding in most of the cases thanks to its quality to price ratio for a given set of features. Accordingly, a number of critical aspects have to be considered, such as the ability to pack as many features as needed in a given product or service while concurrently maintaining high quality, reasonable price, and short time -to- market. Consequently, modeling is becoming a key activity in product development since it enables a controlled design process that can allow for design reuse and easy feature integration in a predictable timeline. Also, the economical globalization and multinational cooperation trends are active drivers pushing for standardized modeling and engineering practices. Standardization could potentially facilitate the deployment and adoption of cost-effective verification and validation methodologies that could, in turn, be harmonized with current business goals.

The overall objective in modern product development for maximizing immediate economical gains can cause conflicts between business goals and engineering practices. Consequently, many aspects such as ergonomics, various environmental concerns, thorough verification and validation, or computational efficiency are frequently ignored. This is especially the case in those areas where the effects are not immediately apparent such as in software applications (e.g., operating systems, web

browsers, office suites) or in software-intensive systems (e.g., computer and mobile networking, portable/wearable electronics, and pervasive computing).

Although we can observe an ever-increasing awareness regarding ergonomics and environmental issues, the same can hardly be said about the necessity for a thorough verification and validation methodology for modern software and software-intensive systems. Nowadays, most of the systems' verification and validation is usually performed through testing and simulation; these techniques although very convenient for certain categories of systems do not provide necessarily for a complete screening of the possible states of a system.

Thus, it becomes apparent that an appropriate science for conscience setting should always accompany, to some extent, any development initiative. Furthermore, in such a setting, it is not so difficult to acknowledge that the most precious resource that human society benefits from is not any physical or material datum but rather the human resources themselves. As such, one may argue that as human society becomes increasingly technology dependant, the importance of assuring robust, bug-free, and high-quality software and systems is also equally increasing.

Systems engineering has marked an impressive comeback in the wake of the new challenges that emerged in this modern era of complex systems design and development. As the main objective of systems engineers is to enable the realization of successful systems, verification and validation represent an important process that is used for the quality assessment of the engineered systems and their compliance with the requirements established at the origin of the system development. Furthermore, in order to cope with the growing complexity of modern software and systems, systems engineering practices have undergone a fundamental transition to a model-based approach. In these settings, systems engineering community and standardization bodies developed interest in using and/or developing some specific modeling language that supports the cooperative work and the exchange of information among systems engineering practitioners.

This volume investigates the available means that can be employed in order to provide a dedicated approach toward the automatic verification and validation of systems engineering design models expressed in standardized modeling languages. It also provides a bird's eye view of the most prominent modeling languages for software and systems engineering, namely the unified modeling language (UML) and the more recent systems modeling language (SysML). Moreover, it innovates with the elaboration of a number of quantitative and qualitative techniques that synergistically combine automatic verification techniques, program analysis, and software engineering quantitative methods, applicable to design models described in modern modeling languages such as UML and SysML. Here is the way, the rest of this book is organized: Chap. 1 presents an introduction to the verification and validation problem, systems engineering, various related standards, and modeling languages along with the model-driven architecture (MDA) concept. In Chap. 2, we discuss the paradigm of architecture frameworks adopted and extended by defense organizations but also employed in the commercial arena as enterprise architecture frameworks. Moreover, in Chap. 3, we provide an overview of the UML 2.0 modeling language in the historical context leading to its emergence. Then, we present in

Chap. 4 the more recent SysML modeling language, the chronological account of its adoption and the commonalities and specific differences between SysML and UML. Chapter 5 describes the verification, validation, and accreditation concepts. Therein, we review noteworthy assessment methodologies based on software engineering techniques, formal verification, and program analysis. Chapter 6 describes an effective and synergistic approach for the verification and validation of systems engineering design models expressed in standardized modeling languages such as UML and SysML. Moreover, Chap. 7 demonstrates the relevance and usefulness of software engineering metrics in assessing structural system aspects captured by UML class and package diagrams. Chapter 8 details the automatic verification and validation of UML behavioral diagrams. Computational models are derived from state machine, sequence, or activity diagrams and are matched against logical properties that capture verification and validation requirements. The corresponding model-checking verification methodology is also described. Chapter 9 discusses the mapping of SysML activity diagrams annotated with probabilities to Markov decision processes (MDP) that can be assessed by probabilistic model checking procedures. Furthermore, Chap. 10 details the performance analysis of SysML activity diagrams annotated with time constraints and probability artifacts using a discrete-time Markov chain model. Chapter 11 is dedicated to the semantic foundations of SysML activity diagrams. We define a probabilistic calculus that we call activity calculus (AC). The latter algebraically captures the formal semantics of the activity diagrams using the operational semantics framework. In Chap. 12, we establish the soundness of the translation of SysML activity diagrams into PRISM specifications. This ensures that the code generated by our algorithm correctly captures the behavior intended by the SysML activity diagram given as input. Finally, a discussion of the presented work together with some concluding remarks are sketched as conclusion in Chap. 13.

Acknowledgments

We would like to express our deepest gratitude to all the people who contributed to the realization of this work. Initially, our research on the verification and validation of systems engineering design models has been supported thanks to a research contract from the Defence Research & Development Canada, the R&D establishment for Canada's Department of National Defence, under the Collaborative Capability Definition, Engineering and Management (Cap-DEM) project. We also acknowledge support from the Natural Sciences and Engineering Research Council (Discovery Grant) of Canada and Concordia University (Research Chair Tier I). We would like also to express our gratitude to the members of the Computer Security Laboratory of Concordia University who helped in reviewing the preliminary versions of this book.

Contents

List of Figures

List of Tables

Acronyms

AP233	application protocol 233
ATM	automated teller machine
CASE	computer-aided software engineering
CBO	coupling between object classes
CCRC	class category relational cohesion
CPU	central processing unit
CR	class responsibility
CTL	computation tree logic
CTS	configuration transition system
DIT	depth of inheritance tree
DMS	distance from the main sequence
DMSO	Defense Modeling and Simulation Organization
DoD	Department of Defense
DoDAF	Department of Defense Architecture Framework
DTMC	discrete-time Markov chain
EFFBD	enhanced function flow block diagram
EIA	Electronic Industries Alliance
FSM	finite state machine
HDL	hardware description language
ICAM	integrated computer-aided manufacturing
IDEF	integrated definition language
IEEE	Institute of Electrical and Electronics Engineers
INCOSE	International Council on Systems Engineering
ISO	International Organization for Standardization
IT	information technology
LCA	least common ancestor
LTL	linear time logic
MDA	model-driven architecture
ML	modeling language
MOF	MetaObject Facility
M&S	modeling and simulation
NAVMSMO	Navy Modeling and Simulation Management Office

NMA	number of methods added
NMI	number of methods inherited
NMO	number of methods overridden
NOA	number of attributes
NOC	number of children
NOM	number of methods
OCL	object constraint language
OMG	Object Management Group
OMT	object modeling technique
PCTL	probabilistic computation tree logic
PDA	personal digital assistant
PMR	public methods ratio
PRISM	probabilistic symbolic model checker
QA	quality assurance
R&D	research and development
RPG	Recommended Practices Guide
SIX	Specialization Index
SOS	structural operational semantics
SoS	systems on top of systems
SMV	symbolic model verifier
SPT	schedulability performance and time
STEP	standard for the exchange of product model data
SysML	system modeling language
TCTL	timed computation tree logic
TeD	telecommunications description language
TOGAF	The Open Group Architecture Framework
UML	unified modeling language
V&V	verification and validation
VV&A	verification, validation, and accreditation
WebML	Web modeling language
XML	extensible markup language