Model-Driven Dependability Assessment of Software Systems

Simona Bernardi • José Merseguer
Dorina Corina Petriu

Model-Driven Dependability Assessment of Software Systems

Springer

Simona Bernardi
Centro Universitario de la Defensa
Academia General Militar
Zaragoza, Spain

José Merseguer
Departamento de Informática
Universidad de Zaragoza
Zaragoza, Spain

Dorina Corina Petriu
Department of Systems and Computer Engineering
Carleton University
Ottawa
Ontario, Canada

ISBN 978-3-642-39511-6 ISBN 978-3-642-39512-3 (eBook)
DOI 10.1007/978-3-642-39512-3
Springer Heidelberg New York Dordrecht London

Preface

Goal of the Book

During the last two decades, a major challenge for the researchers working on modeling and evaluation of computer-based systems has been the assessment of system Non-functional Properties (NFP), such as performance, schedulability, dependability, or security. We can say that this is still an open research challenge today, although considerable progress has been made and different approaches have been developed, which are of interest not only to researchers but also to practitioners in the field.

A class of successful approaches found in the literature relies on building traditional formal models for NFP analysis (such as fault trees, Markov chains, Petri nets, or Bayesian networks) from system descriptions based on the Unified Modeling Language (UML). UML is a widely used modeling language, adopted by both industry and academia, which has become the "lingua franca" for software modeling. Model transformations from UML to formal models are addressed either by providing informal guidelines (implemented, e.g., using Java) or by defining rigorous model transformations (e.g., using QVT 2011; Jouault and Kurtev 2006). The former is included under the umbrella of *model-based development (MBD)* and the second of *model-driven development (MDD)*.

MDD is a software development paradigm characterized by the fact that the primary focus and products of the development are models rather than computer programs. A key premise behind MDD is that the programs are automatically generated from the models. The advantage of MDD is that models are expressed using concepts that are much closer to the problem domain than to the underlying implementation technology, making the models easier to specify, understand, and maintain (Selic 2003).

On the other hand, in model-based development (MBD) the software models do not play the same key role of driving the development process as in MDD, although they still play an important role. For instance, models may be used to guide the

writing of program code by programmers, rather than being used for automatic code generation as in MDD.

Validation of the models is another important aspect, which is addressed using assertions or constraints in MDD, while in MBD validation is usually carried out by hard coded parsers.

Most of the existing approaches for NFP modeling and assessment that are using UML as software modeling language do propose UML extensions for specifying NFPs, which in turn lead to the definition of ad hoc UML profiles for NFP specification and assessment. Such ad hoc profiles usually cover a limited subset of concepts from the NFP domain they are addressing, without any coordination with other similar profiles with respect to the coverage of the NFP domain, terminology consistency, or profile structure.

The situation is better in the performance and schedulability analysis domain, which is supported by two Object Management Group (OMG) standards, "The UML Profile for Schedulability, Performance and Time" (SPT) defined for UML 1.X and "The UML Profile for MARTE: Modeling and Analysis of Real-Time and Embedded Systems", defined for UML 2.X. Many researchers have used SPT or MARTE to add performance and/or schedulability annotations to UML models and then to define model transformations for deriving a variety of performance and/or schedulability models to be used for analysis. Unfortunately, there is no similar standard profile for dependability analysis of UML-based models.

Another OMG standard specifying UML extensions for a variety of non-functional properties, the Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms (QoS&FT), provides a flexible but heavyweight mechanism to define properties such as performance, security, or reliability by means of specific QoS catalogs. The annotation mechanism is supported by a two-step process, which implies catalog binding and either the creation of extra objects just for annotation purposes or the specification of long Object Constraint Language (OCL) expressions. However, this two-step process requires too much effort from the users and may produce models that are hard to understand.

In this context, our book describes the state of the art in modeling and assessment of dependability requirements throughout the software life cycle. Dependability is a term that encompasses several non-functional properties of systems: *availability* (readiness for correct service), *reliability* (continuity of correct service), *safety* (absence of catastrophic consequences for the users and the environment), *integrity* (absence of improper system alterations), and *maintainability* (ability to undergo modification and repairs).

In this book we are using three kinds of models:

(a) *software models* used for software/architecture development and represented in software modeling languages, for instance, UML or AADL;
(b) *software models with dependability annotations* obtained from (a) by adding information related to dependability properties;

- in modeling language allowing for standard extension mechanisms, such as UML, the extra dependability information is given by defining appropriate extensions (i.e., defining a UML profile for dependability);
- in other modeling languages that are not provided with standard extension mechanisms, the language definition must be extended with built-in features for dependability information (such as AADL).

(c) *formal models* such as fault trees, Markov chains, Petri nets, or Bayesian networks, used for analysis; such models are too abstract for software development, but have the advantage of being supported by existing analytic or simulation analysis methods.

The following three research challenges are related to bridging the gap between these three kinds of models. The first challenge, related to dependability modeling, handles the transition from model (a) to (b). In the case of extensible languages (such as UML) this amounts to defining a suitable dependability profile and then applying the profile to model (a) in order to obtain the corresponding model (b). The second challenge, related to dependability analysis, addresses the model transformation from (b) to (c); the actual analysis of model (c) is performed with existing solvers for the formal model used in each case. The last challenge, related to feedback from analysis results to advise for developers on how to improve model (b), relies on bridging the gap between models (c) and (b). The book addresses the first two challenges; in regard to the third one, there are still open research issues that will be discussed in the conclusion.

Emphasis of the Book

In this book, we consider cutting-edge model-driven techniques for modeling and analysis of software dependability, proposed in the last two decades. Most of them are based on the use of UML as software specification language. From the software system specification point of view, such techniques exploit the standard extension mechanisms of UML (i.e., UML profiling). UML profiles enable the software engineers to add dependability non-functional properties to the software model, besides the functional ones.

The book presents the state of the art on UML profile proposals for dependability specification and rigorously describes the trade-off they accomplish. The focus is mainly on the RAMS (reliability, availability, maintainability, and safety) properties. Among the existing profiles, we emphasize the DAM (Dependability Analysis and Modeling) profile, which attempts to unify, under a common umbrella, the previous UML profiles from literature providing capabilities for dependability specification and analysis. DAM is defined as an extension of the MARTE profile and reuses some of its definitions.

Another concern addressed in the book is the assessment of system dependability, which has been traditionally carried out using standard modeling techniques, some

based on the use of formal models (e.g., fault frees for reliability analysis, stochastic Petri Nets for both reliability and availability analysis). We particularly address the construction of such formal analysis models by model-to-model transformation techniques that, given a UML software model enriched with dependability annotations, produce either automatically or systematically proper dependability analysis models. These models can be analyzed with known solution techniques specific to the formalism used by the respective models and supported by existing software tools. The book describes two prominent model-to-model transformation techniques, proposed in the literature, that support the generation of the analysis model and allow for further assessment of different RAMS properties. Case studies from different domains will also be presented, in order to provide examples for practitioners about how to apply the aforementioned techniques.

Target Audience

The book is research oriented and it is mainly addressed to students taking Master and PhD courses in software engineering. They will learn the basic dependability concepts and how to model them, using the current de facto standard UML modeling language and its extension by the profiling approach. In particular, the students will be able to apply the UML extensions defined by the DAM profile and the standard MARTE profile, on which DAM is based. They will also gain insight into dependability analysis techniques, through the use of appropriate modeling formalisms, as well as of model-to-model transformation techniques for deriving dependability analysis models from UML specifications. The book provides proper references for further readings on these topics and a discussion on open issues in this research area.

Moreover, software practitioners interested in dependability analysis are also a target audience for this book. They will find a unified framework for the specification of dependability requirements and properties with UML that can be used throughout the entire software life cycle. They will also learn, with the help of the proposed case studies, rigorous techniques for deriving different dependability analysis models, such as fault trees to Petri nets, from UML software models with dependability annotations.

Road Map of the Book

Chapter 1

Establishes the scope, objectives, and point of view of the book with respect to model-driven software dependability specification and assessment.

Chapter 2

Presents the main dependability concepts used throughout the book. The chapter is mainly addressed to beginners in the dependability field, since it provides useful references for creating a background in dependability. Readers already familiar with dependability could skip the chapter.

Chapter 3

Model-driven software dependability assessment, the topic of the book, is necessarily carried out using models, in particular software models. Therefore, readers need to know what languages features support software modeling for dependability specification and assessment. This chapter is devoted to both general purpose modeling language, such as UML, and Domain-Specific Modeling Languages (DSMLs) used in literature for dependability assessment. It describes first the use of UML diagrams for software modeling, with emphasis on diagrams used for dependability modeling and assessment, even though concrete proposals are not described yet. An important characteristic of UML is discussed next: its profiling mechanism as a technique to define DSMLs as lightweight extensions of UML. Although the focus of the book is on UML, the chapter briefly presents another DSML, concretely AADL, so that the reader can see an approach to dependability modeling and analysis that is different from those based on UML.

Chapter 4

Irrespective of the approach taken for defining a DSML, the domain model is usually the first step toward the DSML definition. A domain model, described as a UML class diagram, specifies a set of core concepts related to a specific problem or field. This chapter introduces a domain model for dependability characteristics, aimed at both dependability modeling and analysis. The dependability concepts, given in Chap. 2, constitute the basis for this domain model, which unifies the terminology used in previous dependability profiles proposed in the literature and provides a consistent vocabulary for software dependability modeling and analysis.

Chapter 5

Based on the domain model presented in Chap. 4, this chapter develops a UML profile for dependability modeling and analysis of software systems. The profile, called DAM, relies on the standard OMG MARTE (described in Appendix A). DAM consists of a set of UML extensions (i.e., stereotypes, tag values, and constraints) to annotate a UML model with dependability properties, requirements, and measures for dependability analysis purposes. To exemplify the use of DAM, the chapter

applies the profile to two case studies. The first one is in the field of secure distributed systems and the second in the avionics field. Later, in Chap. 8, these case studies will be used for the illustration of dependability analysis.

Chapter 6

Dependability modeling has been the topic developed in the book so far. This chapter introduces dependability analysis concerns. Dependability analysis is carried out using either formal models or systematic methods, and this chapter provides an overview of the most common ones, which are compliant with current industrial standards (i.e., the International Electrotechnical Commission standards). Special attention is given to Fault Tree and Stochastic Petri Net formalisms, since they will be used later in Chap. 8 for the dependability analysis of the case studies.

Chapter 7

During the last two decades, several proposals have been developed to create dependability DSMLs (D-DSMLs). DAM, presented in Chap. 5, is an example of a D-DSML developed as a UML profile. Most of these proposals also accomplish the transformation of the D-DSML into proper dependability analysis models, as those from Chap. 6. This chapter presents and evaluates 36 proposals from the literature, most of them having in common that their D-DSML is based on UML.

Chapter 8

The objective of this chapter is to describe some proposals of interest for practitioners, selected from those presented in Chap. 7. The focus of interest is on how these proposals address the translation of a D-DSML into models for analysis. Concretely, the chapter focuses on availability and reliability proposals. We selected one from Bernardi et al., addressing availability, and the another from Pai and Dugan, addressing reliability. These two approaches are applied to the case studies developed in Chap. 5. Availability analysis is then applied to a secure distributed system case study, while reliability models are obtained for a mission avionics case study.

Chapter 9

Once the state of the art on dependability modeling and analysis of software systems has been presented, the last chapter discusses research issues that are still open and need additional effort.

Acknowledgments

Authors would like to thank Vittorio Cortellessa and Julio Medina, whose comments helped to substantially improve the book. Special thanks are due to the editor, Ralf Gerstner, for his advice in structuring the book and for his patience.

Zaragoza, Spain
Ottawa, Canada
2013

Simona Bernardi, José Merseguer
Dorina Corina Petriu

Contents

Acronyms

AADL	Architecture Analysis and Design Language
BPEL	Business Process Execution Language
BPMN	Business Process Modeling Notation
CTMC	Continuous Time Markov Chain
CGSPN	Concurrent Generalized Stochastic Petri Net
DAM	Dependability Modeling and Analysis
DFT	Dynamic Fault Tree
DSL	Domain-Specific Language
DSML	Domain-Specific Modeling Language
D-DSML	Dependability-DSML
DSPN	Deterministic and Stochastic Petri Net
ESPN	Extended Stochastic Petri Net
FFA	Functional Failure Analysis
FMEA	Failure Mode and Effect Analysis
FMECA	Failure Mode, Effect, and Criticality Analysis
FT	Fault Tree
GQAM	General Quantitative Analysis Model
GSPN	Generalized Stochastic Petri Net
HAZOP	HAZard and OPerability studies
IEC	International Electrotechnical Commission
MARTE	Modelling and Analysis of Real-Time Embedded systems
M2M	Model-to-Model
MCS	Minimal Cut Set
MDD	Model-Driven Development
MOF	Meta-Object Facility
MRSPN	Markov Regenerative Stochastic Petri Net
NFP	Non-Functional Property
OCL	Object Constraint Language
OMG	Object Management Group
PHA	Preliminary Hazard Analysis
PN	Petri Net

QoS&FT	UML profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms
RAMS	Reliability, Availability, Maintainability, Safety
RBD	Reliability Block Diagram
SAE	Society for Automotive Engineers
SAN	Stochastic Activity Network
SoaML	Service oriented architecture Modeling Language
SPN	Stochastic Petri Net
SPT	UML profile for Schedulabibity, Performance and Time Specification
SRN	Stochastic Reward Net
SWN	Stochastic Well-formed Net
TPN	Time Petri Net
UML	Unified Modeling Language
SysML	Systems Modeling Language
VSL	Value Specification Language