Automated Validation & Verification of UML/OCL Models Using Satisfiability Solvers

Nils Przigoda • Robert Wille
Judith Przigoda • Rolf Drechsler

Automated Validation & Verification of UML/OCL Models Using Satisfiability Solvers

Springer

Nils Przigoda
Mobility Division
Siemens AG
Braunschweig, Germany

Judith Przigoda
University of Bremen
Bremen, Germany

Robert Wille
Johannes Kepler University Linz
Linz, Austria

Rolf Drechsler
AG Rechnerarchitektur
University of Bremen
Bremen, Germany

Cyber-Physical Systems
DFKI GmbH, Bremen, Germany

ISBN 978-3-319-72813-1 ISBN 978-3-319-72814-8 (eBook)
https://doi.org/10.1007/978-3-319-72814-8

Library of Congress Control Number: 2017961733

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

Preface

Only four decades after the first manned flight to the moon, a common device such as a smartphone consists of more complex technology than the "high-end" computers that controlled the Apollo 11 space mission. The same holds for software which became more complex at an even higher rate. Consequently, the design of such systems became a tremendously hard, difficult, and expensive problem. To cope with this, modeling languages such as UML and SysML were introduced as description means to describe quasi-blueprints in early stages of the design flow. These modeling languages hide implementation details while providing a formal base for the first system analysis.

However, in order to benefit from this abstraction, it has to be ensured that the resulting models are applicable (motivating validation) and correct (motivating verification). But even at this high level of abstraction, this remains a complex problem. Accordingly, researchers heavily investigated the validation and verification of UML/OCL models as well as models described in similar languages. Solutions which are based on so-called satisfiability solvers find particular interests due to their capability to completely cover large search spaces in a—considering the usually exponential complexity—rather efficient fashion.

This book provides a comprehensive description of such methods and their application—including a general flow that utilizes a formalization of UML/OCL. While the presented flow focuses on using satisfiability solvers, how the provided descriptions can additionally be used for any other automatic reasoning engine is also described. Furthermore, for a broad variety of validation and verification tasks, the application of the proposed flow is described. Additionally, the book also briefly covers how nonfunctional properties such as timing constraints can be handled within the described flow.

A case study demonstrates the possibilities and applicability of the presented approaches and shows that there is still a gap between the UML/OCL models and the following design steps. In order to address this problem, finally, an approach is presented which verifies an implementation against its model. This enables the designer to transfer validation and verification results to the following lower abstraction levels.

This book is the result of several years of intensive research at the University of Bremen, Germany; DFKI GmbH Bremen, Germany; the Johannes Kepler University Linz, Austria; and, recently, Siemens AG in Braunschweig, Germany. During this time, we experienced broad support from many people for which we would like to thank them very much. Most importantly, we are thankful to the respective groups in Bremen, Linz, and Braunschweig for providing a comfortable and inspirational environment from which some authors benefit until today. Particular thanks go to (in alphabetical order) Christoph Hilken, Frank Hilken, Jannis Stoppe, Jan Peleska, Jonas Gomes Filho, Julia Seiter, Martin Gogolla, Mathias Soeken, Pablo González de Aledo, Pablo Sánchez Espeso, Philipp Niemann, and Ulrich Kühne for the very productive collaboration that resulted in research papers which are partially covered in this book. With respect to funding, we are indebted to thank the German Research Foundation (DFG) which supported the research summarized in this book through the Reinhart Koselleck project under grant no. DR 287/23-1, the Graduate School SyDe funded by the German Excellence Initiative within the University of Bremen's institutional strategy, and the German Ministry of Education and Research (BMBF) for their support through the projects SPECifIC under grant no. 01IW1300 and SELFIE under grant no. 01IW16001. Besides that, the Siemens AG Mobility Division sponsored a scholarship for Nils Przigoda's PhD thesis leading to several results which formed the basis for this book. Finally, we would like to thank Springer and, in particular, Charles "Chuck" Glaser, for making this book possible.

Braunschweig, Germany — Nils Przigoda
Linz, Austria — Robert Wille
Bremen, Germany — Judith Przigoda
Bremen, Germany — Rolf Drechsler
October 2017

Contents

Nomenclature

$\mathbb{B}$	$\mathbb{B} = \{\texttt{true}, \texttt{false}\}$
$\mathbb{N}$	$\mathbb{N} = \{0, 1, 2, \ldots\}$
$\mathbb{Z}$	$\mathbb{Z} = \{\ldots, -1, 0, 1, 2, \ldots\}$
ϵ	Symbol for `null` in OCL
$\perp$	Symbol for `invalid` in OCL
α_a^{υ}	Variable for the attribute a of the object υ in the symbolic representation
β_c	A variable indicating if objects of a class c exist or not
$\boldsymbol{\omega}_{\sigma \to \sigma'}$	Variable for the operation to be executed during the transition from the system state σ to σ'.
δ_a^{υ}	Definedness of an attribute a of the object υ in the symbolic representation
$[\![i/I]\!]$	A transformed OCL expression i or a set of OCL expressions I
Λ	A set of links
λ	A single link, i.e., an instance of an association r
$\lambda_{role_{c_2}}^{\upsilon}$	Variable for possible links of $role_{c_2}$ of the object υ in the symbolic representation
$\mathscr{E}$	The set of all enumerations
$\mathfrak{V}$	The set of all variables of all types
$\mathfrak{V}_t$	The set of all variables with the type t
ω	An operation call $\omega = (\upsilon, o)$
$\Omega_{m,\sigma}$	The set of all operation calls
$\prec, \preccurlyeq$	Inheritance relation between two classes
ρ	The Greek letter ρ is used for a transformed value of a literal or the result a transformed subexpression—in both cases combined a δ-variable to ensure the pair notation (ρ, δ)
σ	A single system state of a model m
Σ_m	The set of all possible system states of a model m
$\dot{\llcorner\!\lrcorner}$	The underbracket is used in SMT-LIB listing to show that term above must be replaced by a precise value or number
A	Attributes of a class
C	The finite set of classes of a model m

c	A single class $c = (A, O, I)$
F	A finite set of frame conditions of an operation o
I	Invariants of a class
m	A (UML/OCL) model $m = (C, R)$
O	Operations of a class
o	A single operation of class $o = (P, r, \lhd, \rhd)$
P	A maybe empty set of parameters of an operation o
R	The finite set of associations between the classes of a model m
r	A single association $r = (role_{c_1} : c_1, role_{c_2} : c_2, (l_1, u_1), (l_2, u_2))$
r	Return value of an operation o
$v : \mathfrak{t}$	A variable with identifier v and type $\mathfrak{t} \in \mathfrak{T}$
$\lhd$	A may empty set of preconditions of an operation o
$\mathscr{C}$	The (infinite) set of all classes
$\mathscr{R}$	The (infinite) set of all associations also called relations
$\mathfrak{T}$	Type system
$\rhd$	A may empty set of postconditions of an operation o
υ	An object instance of a class c
Υ	A set of object instances