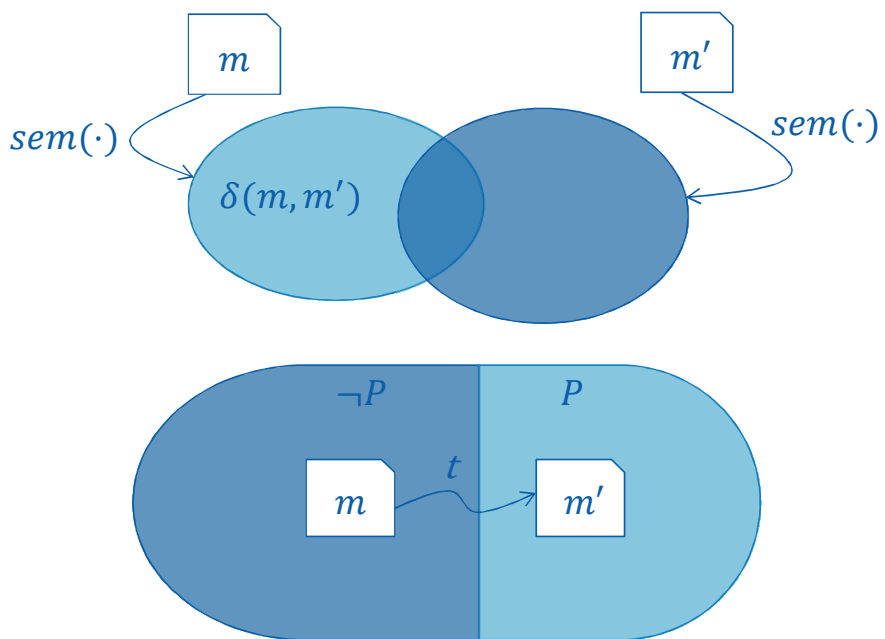


Oliver Kautz

Model Analyses Based on Semantic Differencing and Automatic Model Repair



Model Analyses Based on Semantic Differencing and Automatic Model Repair

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

**M.Sc. RWTH
Oliver Kautz**

aus

Menden (Sauerland), Deutschland

Berichter: Universitätsprofessor Dr. rer. nat. Bernhard Rumpe
University Professor Shahar Maoz, PhD

Tag der mündlichen Prüfung: 10. Februar 2021

D 82 (Diss. RWTH Aachen University, 2021)

Aachener Informatik-Berichte, Software Engineering

herausgegeben von
Prof. Dr. rer. nat. Bernhard Rumpe
Software Engineering
RWTH Aachen University

Band 46

Oliver Kautz
RWTH Aachen University

**Model Analyses Based on Semantic Differencing
and Automatic Model Repair**

Shaker Verlag
Düren 2021

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <http://dnb.d-nb.de>.

Zugl.: D 82 (Diss. RWTH Aachen University, 2021)

Copyright Shaker Verlag 2021

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the publishers.

Printed in Germany.

ISBN 978-3-8440-7926-5

ISSN 1869-9170

Shaker Verlag GmbH • Am Langen Graben 15a • 52353 Düren

Phone: 0049/2421/99011-0 • Telefax: 0049/2421/99011-9

Internet: www.shaker.de • e-mail: info@shaker.de

Eidesstattliche Erklärung

I, Oliver Kautz

erklärt hiermit, dass diese Dissertation und die darin dargelegten Inhalte die eigenen sind und selbstständig, als Ergebnis der eigenen originären Forschung, generiert wurden.

Hiermit erkläre ich an Eides statt

1. Diese Arbeit wurde vollständig oder größtenteils in der Phase als Doktorand dieser Fakultät und Universität angefertigt;
2. Sofern irgendein Bestandteil dieser Dissertation zuvor für einen akademischen Abschluss oder eine andere Qualifikation an dieser oder einer anderen Institution verwendet wurde, wurde dies klar angezeigt;
3. Wenn immer andere eigene- oder Veröffentlichungen Dritter herangezogen wurden, wurden diese klar benannt;
4. Wenn aus anderen eigenen- oder Veröffentlichungen Dritter zitiert wurde, wurde stets die Quelle hierfür angegeben. Diese Dissertation ist vollständig meine eigene Arbeit, mit der Ausnahme solcher Zitate;
5. Alle wesentlichen Quellen von Unterstützung wurden benannt;
6. Wenn immer ein Teil dieser Dissertation auf der Zusammenarbeit mit anderen basiert, wurde von mir klar gekennzeichnet, was von anderen und was von mir selbst erarbeitet wurde;
7. Teile dieser Arbeit wurden zuvor veröffentlicht und zwar in: [BKRW17, BKRW19, DKMR19, KR18a, KR18b, DKMR20].

Abstract

Models are the primary development artifacts used in model-driven software development. Therefore, models continuously evolve during the design, development, and maintenance of software systems. Thus, model differencing is an important task to understand the syntactic and semantic differences between model versions.

Previous work produced general (and thus language-independent) approaches for syntactic model differencing, but only a few language-dependent approaches for semantic model differencing. Approaches combining syntactic with semantic model differencing by relating the syntactic changes of models to their semantic differences rarely exist. Previous work neglected the development of language-independent approaches abstracting from a concrete model property for detecting the syntactic elements of a model, which cause that the model does not satisfy the property. If the property encodes a requirement and the non-satisfaction represents the existence of a bug, then detecting the syntactic model elements causing the non-satisfaction of the property facilitates developers in detecting the syntactic model elements causing the bug.

This thesis presents a framework for precisely defining modeling languages, including syntax, semantics, and model evolution possibilities. To demonstrate its feasibility, the framework is instantiated with four concrete modeling languages: Time-synchronous port automata, feature diagrams, sequence diagrams, and activity diagrams. For each of these modeling languages, this thesis presents syntactic and semantic differencing operators. The operators facilitate developers in understanding the syntactic and semantic differences between models of the languages. Based on the framework for precisely defining modeling languages, this thesis presents a modeling language and property-independent framework for automatic model repairs. The framework facilitates developers in detecting the syntactic elements of a model causing that the model does not satisfy a property. Instantiating the framework with a concrete modeling language and a concrete model property enables the automatic calculation of syntactic changes that transform a model not satisfying the property to a model that satisfies the property. The syntactic model elements affected by the syntactic changes can be interpreted to cause the non-satisfaction of the property. Developers can review the affected elements as evidence for the identification of the required changes for fixing the bug that causes the non-satisfaction of the property. Alternatively, the automatically calculated syntactic changes can be directly applied to the model to obtain a model that satisfies the property. The framework relies on the assumption that it is possible to partition the syntactic changes applicable to each model into finitely many model-specific and property-specific equivalence classes.

This thesis presents formal proofs for the correctness of the language-independent and language-dependent results. The applicability and usefulness of the modeling language-independent frameworks are demonstrated by instantiating the frameworks with four modeling languages and the properties refinement, generalization, and refactoring.

The instantiations of the frameworks with the four modeling languages and the example properties can be directly employed in development processes. The process of instantiating the frameworks is a methodology for the development of syntactic and semantic differencing procedures as well as precise model evolution analyses for detecting syntactic model elements causing the non-satisfaction of properties.

Danksagung

Zunächst danke ich meinem Doktorvater Prof. Dr. Bernhard Rumpe für die Möglichkeit zur Promotion am Lehrstuhl für Software Engineering. Die zahlreichen Diskussionen über die Inhalte dieser Arbeit und andere Themen waren sehr lehrreich und motivierend. Außerdem bin ich sehr dankbar für die gewährten thematischen Freiheiten bezüglich der Inhalte dieser Arbeit. Ich möchte mich auch für die Möglichkeit der Bearbeitung anderer, akademischer und industrienaher, Projekte neben dieser Dissertation bedanken.

Des Weiteren möchte ich Prof. Shahar Maoz, PhD, für die Zweitbegutachtung dieser Arbeit danken. Ich danke außerdem Prof. Dr. Klaus Wehrle für die Leitung des Prüfungskomitees und Prof. Dr. Erika Ábrahám für die Abnahme der Prüfung im Bereich der theoretischen Informatik.

Bedanken möchte ich mich außerdem bei meinen weiteren, teilweise ehemaligen, Kolleginnen und Kollegen für die schöne und erfolgreiche Zeit am Lehrstuhl. Diesbezüglich gilt mein Dank Jun.-Prof. Dr. Andreas Wortmann, Dr. Katrin Hölldobler, Dr. Judith Michael, Vincent Bertram, Arvid Butting, Joel Charles, Manuela Dalibor, Imke Drave, Robert Eikermann, Arkadii Gerasimov, Steffen Hillemacher, Nico Jansen, Jörg Christian Kirchof, Evgeny Kusmenko, Achim Lindt, Matthias Markthaler, Lukas Netz, Deni Raco, David Schmalzing, Sebastian Stüber, Simon Varga, Louis Wachtmeister, Vassily Aliseyko, Marita Breuer, Niklas Dienstknecht, Christoph Engels, Joshua Mingers, Brian Sinkovec, Nina Pichler, Annika Donath, Galina Volkova, Lennart Bucher, Jerome Pfeiffer, Sylvia Gunder, Sonja Müßigbrodt, Dr. Timo Greifenberg, Dr. Markus Look, Dr. Klaus Müller, Dr. Pedram Mir Seyed Nazari, Dr. Christoph Schulze und Dr. Michael von Wenckstern. Für das Korrekturlesen früherer Fassungen dieser Arbeit bedanke ich mich bei Andreas, Arvid, David, Evgeny, Imke, Judith, Katrin, Nico, Robert, Sebastian, Simon und Steffen.

Ich möchte mich auch bei meiner Familie und meinen Freunden bedanken. Mein besonderer Dank gilt meinen Eltern Karin und Joachim sowie meiner Schwester Sabrina für die kontinuierliche Unterstützung meiner Vorhaben, die zu Anfertigung dieser Arbeit geführt haben. Zusätzlich bedanke ich mich bei Theodor, Hildegard, Till, Pia und Steffen. Auf euer Interesse und eure motivierende Unterstützung war immer Verlass.

Zuletzt bedanke ich mich von ganzem Herzen bei meiner Ehefrau Romina. Ich konnte mich immer auf deine volle Unterstützung verlassen, du hast mich immer motiviert und einige Opfer in unserem Privatleben erbracht, sodass ich mich bestmöglich meinem Promotionsvorhaben widmen konnte. Dafür bin ich dir zutiefst dankbar.

Contents

I	Prologue	1
1	Introduction	3
1.1	Context of the Thesis	5
1.2	Main Goals and Contribution	6
1.3	Thesis Organization	6
1.4	Notational Conventions and Mathematical Foundations	7
1.4.1	Sets and Functions	7
1.4.2	Finite and Infinite Words	8
1.4.3	Countable Sets	9
1.4.4	Nondeterministic Finite Automata	9
1.4.5	Büchi Automata	10
1.4.6	Graphs and Trees	11
1.5	Own Related Publications	11
2	A Generic Framework for Defining Modeling Languages	13
2.1	Modeling Language	14
2.2	Change Operations and Syntactic Differencing	19
2.3	Universe of Names	22
2.4	A Template for Describing Change Operations	22
2.5	Related Work	24
2.5.1	Modeling Language Definition and Variability	24
2.5.2	Syntactic Model Differencing and Change Operations	25
2.5.3	Semantic Model Differencing	26
II	Concrete Instantiations of the Generic Framework	31
3	Finite Time-Synchronous Port Automata	33
3.1	Time-synchronous Port Automata Syntax	35
3.2	Time-synchronous Port Automata Semantics	36
3.3	Semantic Differencing of Time-synchronous Port Automata	38
3.4	Time-synchronous Port Automata Change Operations	43
3.4.1	State-Addition Operations	44

3.4.2	State-Deletion Operations	45
3.4.3	Transition-Addition Operations	46
3.4.4	Transition-Deletion Operations	47
3.4.5	Input-Channel-Addition Operations	49
3.4.6	Output-Channel-Addition Operations	50
3.4.7	Channel-Deletion Operations	51
3.4.8	Initial-State-Change Operations	52
3.5	Time-Synchronous Port Automaton Modeling Language	53
3.6	Related Work	54
4	Feature Diagrams	57
4.1	Feature Diagram Syntax	58
4.2	Feature Diagram Semantics	60
4.3	Semantic Differencing of Feature Diagrams	63
4.4	Feature Diagram Change Operations	71
4.4.1	Feature-Addition Operations	72
4.4.2	Feature-Deletion Operations	73
4.4.3	Implies-Constraint-Addition Operations	74
4.4.4	Implies-Constraint-Deletion Operations	75
4.4.5	Excludes-Constraint-Addition Operations	76
4.4.6	Excludes-Constraint-Deletion Operations	77
4.4.7	Or-Group-Creation Operations	78
4.4.8	Xor-to-Or-Conversion Operations	79
4.4.9	Or-to-Xor-Conversion Operations	80
4.4.10	Mandatory-to-Optional-Conversion Operations	81
4.4.11	Optional-to-Mandatory-Conversion Operations	82
4.4.12	Feature-Group-Insertion Operations	83
4.4.13	Feature-Group-Exclusion Operations	85
4.4.14	Root-Rename Operations	85
4.5	Feature Diagram Modeling Language	86
4.6	Related Work and Discussion	88
5	Sequence Diagrams	91
5.1	Sequence Diagram Syntax	93
5.2	Sequence Diagram Semantics	95
5.3	Semantic Differencing of Sequence Diagrams	97
5.4	Sequence Diagram Change Operations	107
5.4.1	Object-Addition Operations	109
5.4.2	Object-Deletion Operations	110
5.4.3	Tag-Object-as-Complete Operations	112
5.4.4	Untag-Object-as-Complete Operations	113

5.4.5	Tag-Object-as-Visible Operations	114
5.4.6	Untag-Object-as-Visible Operations	115
5.4.7	Tag-Object-as-Initial Operations	116
5.4.8	Untag-Object-as-Initial Operations	118
5.4.9	Action-Addition Operations	119
5.4.10	Action-Deletion Operations	120
5.4.11	Interaction-Addition Operations	121
5.4.12	Interaction-Deletion Operations	122
5.5	Sequence Diagram Modeling Language	123
5.6	Related Work	124

6 Activity Diagrams 129

6.1	Activity Graph Syntax	131
6.2	Activity Graph Trace Semantics	133
6.3	Semantic Differencing of Activity Graphs	137
6.4	Activity Graph Change Operations	139
6.4.1	Label-Addition Operations	142
6.4.2	Label-Deletion Operations	143
6.4.3	Action-Insertion Operations	144
6.4.4	Action-Deletion Operations	144
6.4.5	Xor-Fragment-Insertion Operations	145
6.4.6	Xor-Fragment-Deletion Operations	146
6.4.7	And-Fragment-Insertion Operations	147
6.4.8	And-Fragment-Deletion Operations	148
6.4.9	Cyclic-Fragment-Insertion Operations	149
6.4.10	Cyclic-Fragment-Deletion Operations	150
6.4.11	Fragment-Branch-Insertion Operations	152
6.4.12	Fragment-Branch-Deletion Operations	152
6.5	Activity Diagram Modeling Language	154
6.6	Related Work	156

III Automatic Model Repairs 159

7 A Framework for Automatic Model Repairs 161

7.1	Motivating Examples in Context of Repairing Refinement	163
7.1.1	Shortest Repair of a Failed Activity Diagram Refinement Step	164
7.1.2	Shortest Repair of a Time-Synchronous Port Automaton to Achieve the Satisfaction of a Requirement	166
7.1.3	Understanding a Feature Diagram Evolution Step	168
7.1.4	Understanding the Semantic Differences between Sequence Diagrams	169

7.2	Model Repair Problems	171
7.3	Change Operation Properties	176
7.4	Computing Shortest Repairing Change Sequences	183
7.5	Algorithms for Computing Shortest Solutions	189
7.5.1	Algorithm Performance	191
7.5.2	Checking Change Operation Properties	192
7.5.3	Propagating Properties Implying the Complement Model Property	194
7.5.4	Detecting Previously Explored Models	196
7.6	Applicability and Development Methodology	200
7.7	Composing Model Repair Problems	202
7.7.1	Derivation of Operations that Delay Solutions	203
7.7.2	Derivation of Operations that Induce Equally Long Shortest Solutions	206
7.7.3	Repair-Representative Function Derivation	209
7.8	Related Work and Discussion	211

8 Concrete Instantiations of the Model Repair Framework 215

8.1	Refines, Generalizes, and Refactors Model Repair Problems	215
8.2	Instantiations with the Time-Synchronous Port Automaton Language	217
8.2.1	Time-Synchronous Port Automaton Refinement Repair	218
8.2.2	Time-Synchronous Port Automaton Generalization Repair	220
8.2.3	Time-Synchronous Port Automaton Refactoring Repair	221
8.2.4	Repair-Representative Function and Example Applications	222
8.2.5	Implementation and Experiments	223
8.3	Instantiations with the Feature Diagram Language	231
8.3.1	Feature Diagram Refinement Repair	232
8.3.2	Feature Diagram Generalization Repair	234
8.3.3	Feature Diagram Refactoring Repair	236
8.3.4	Example Repair-Representative Function and Application	236
8.3.5	Implementation and Experiments	237
8.4	Instantiations with the Sequence Diagram Language	245
8.4.1	Sequence Diagram Refinement Repair	246
8.4.2	Sequence Diagram Generalization Repair	249
8.4.3	Sequence Diagram Refactoring Repair	251
8.4.4	Repair-Representative Function and Example Applications	252
8.4.5	Implementation and Experiments	253
8.5	Instantiations with the Activity Diagram Language	257
8.5.1	Activity Diagram Refinement Repair	259
8.5.2	Activity Diagram Generalization Repair	262
8.5.3	Activity Diagram Refactoring Repair	264
8.5.4	Repair-Representative Function and Example Applications	265

8.5.5	Implementation and Experiments	266
IV	Epilogue	273
9	Conclusion and Future Work	275
9.1	Summary and Main Results	275
9.2	Possible Future Work Directions	279
	Bibliography	281
A	Time-Synchronous Port Automata for Experimental Evaluations	301
B	Feature Diagrams for Experimental Evaluations	303
C	Sequence Diagrams for Experimental Evaluations	307
D	Activity Diagrams for Experimental Evaluations	309
	List of Figures	313
	Acronyms	319
	Glossary of Notation for Foundations	321
	General Glossary of Notation	323
	Index	329