Informatik

Anne-Thérèse Körtgen

Modellierung und Realisierung von Konsistenzsicherungs-werkzeugen für simultane Dokumentenentwicklung

SHAKER VERLAG

Modellierung und Realisierung von Konsistenzsicherungswerkzeugen für simultane Dokumentenentwicklung

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

vorgelegt von

Diplom-Informatikerin
Anne-Thérèse Körtgen

aus Bonn-Beuel

Berichter: Universitätsprofessor Dr.-Ing. Manfred Nagl
Universitätsprofessor Dr. rer. nat. Wilhelm Schäfer

Tag der mündlichen Prüfung: 25. Juni 2009

[Kör09] A. Körtgen
Modeling and Realization of Consistency Maintaining Tools for Simultaneous Document Development
Dissertation Thesis, in German, 289 pp., RWTH Aachen, Shaker Verlag, 2009.
www.se-rwth.de/publications

Zusammenfassung

Die Ergebnisse von Entwicklungsprozessen werden in Dokumenten festgehalten. Diese enthalten Spezifikationen/Realisierungen des zu entwickelnden Systems aus unterschiedlichen Blickwinkeln und auf verschiedenen Abstraktionsstufen. Dokumente können sich inhaltlich überlappen, wodurch es bei paralleler Bearbeitung der Dokumente zu inkonsistenten, d. h. widersprüchlichen Beschreibungen kommen kann.

Thema dieser Arbeit sind allgemeine Konzepte und Werkzeuge zur Konsistenzwiederherstellung, auch *Integration* genannt, wobei die syntaktische Konsistenz zwischen Dokumenten verschiedener Abstraktionsstufen im Vordergrund steht. Die Arbeit baut auf den im SFB IMPROVE erzielten Ergebnissen zu Integrationswerkzeugen auf. Zur Demonstration der Anwendbarkeit der eingeführten Konzepte werden insbesondere Konsistenzprobleme in verfahrenstechnischen Entwicklungsprozessen und zwischen den dort entwickelten Fließbildern untersucht.

Bestehende Ansätze lösen Inkonsistenzen durch Transformation auf, die meisten Ansätze, wie auch der Ansatz dieser Arbeit, verwenden Modell- und Graphtransformationen, wobei sie feingranulare Beziehungen überlappender Strukturen zwischen Dokumenten speichern. Die Vorgehensweisen zur Inkonsistenzbehebung sind jedoch nicht ausreichend.

Des Weiteren sind die verwendeten Transformationssprachen für umfangreiche, komplexe Dokumente nicht ausdrucksstark genug. Es bedarf einiger Modularisierungs- und Operationalisierungskonzepte sowie weiterer Ausdrucksmittel zur Spezifikation komplexer Graphstrukturen. Zudem berücksichtigen sie für die Erzeugung von Dokumentelementen nicht das Layout der Dokumente und das in den Graphmustern der Transformationen angegebene Layout, welches insbesondere in verfahrenstechnischen Fließbildern sehr wichtig ist.

Die vorliegende Arbeit liefert allgemeine Konzepte zur Konsistenzwiederherstellung und demonstriert die Umsetzbarkeit mit einer prototypischen Realisierung an Beispielen aus der Verfahrenstechnik und der Softwareentwicklung. Verschiedenartige, konsistenzwiederherstellende Transformationen, *Reparaturaktionen* genannt, werden dynamisch zur Laufzeit erzeugt unter Berücksichtigung der inkonsistenten Beziehungen zweier Dokumente und der dort auftretenden Fehler. Vom Benutzer durchgeführte komplizierte Restrukturierungen an einem Dokument werden durch die dynamische Behandlung erkannt. Die erstellten Transformationen definieren verschiedene Alternativen, wie sich Änderungen des Benutzers auf beide Dokumente auswirken können (u. U. an beiden Dokumenten gleichzeitig), um einen konsistenten Zustand zu erreichen. Die Alternativen werden dem Benutzer präsentiert, der durch die Auswahl einer Transformation bestimmt, wie die Inkonsistenz behoben werden soll.

Die zur Wiederherstellung der Konsistenz verwendete Graphtransformationssprache wurde modularisiert, d. h. Graphmuster- und Transformationsspezifikationen können in andere Spezifikationen zur Wiederverwendung importiert werden. Ent-

weder werden sie ganzheitlich importiert, um diese zu erweitern, oder anteilig, um die Anteile in der linken Regelseite einer Transformation als sog. *Kontext* zu verwenden. Es entstehen zwischen Transformationen je nach Verwendungsart *Erweiterungs-* oder *Kontextbeziehungen*.

Die Erweiterungsbeziehungen werden insbesondere bei der Ermittlung geeigneter Reparaturaktionen berücksichtigt, um ggf. noch konsistente Korrespondenzbeziehungen bei *kleineren* zu verwendenden Transformationen zu erhalten. Kontextbeziehungen zwischen Transformationen optimieren die Mustersuche und ermöglichen Abhängigkeitsanalysen zwischen ausführbaren Transformationen. Diese Abhängigkeiten sind für den Benutzer bei einer Transformationsentscheidung nützlich.

Des Weiteren erlaubt die Transformationssprache operationale Aufrufe anderer Transformationen mit imperativen Sprachelementen. Es wurden außerdem Ausdrucksmittel eingeführt, um mengenwertige, wiederholte und alternative Graphmuster zu spezifizieren, und Ausdrucksmittel, um ähnliche Elemente mit Stringmatching-Techniken und Synonymvergleichen aufzuspüren.

Die Erstellung der Transformationen ist ein aufwändiger Prozess. Zur Unterstützung dieses Prozesses sind in dieser Arbeit zwei Ansätze entwickelt worden, in denen aus bestehenden Korrespondenzbeziehungen (i) zwischen Elementen der Dokumentenebene bzw. (ii) zwischen Typen und Attributen der Dokumentenmodellebene Transformationen induziert werden. Die Korrespondenzbeziehungen werden zuvor mit einer interaktiven Korrespondenzanalyse ermittelt, die auch von dem Konsistenzsicherungswerkzeug durchgeführt wird.

Die Analysen finden - jeweils passend zu einem generischen Regelsatz, der nur *grobe* Korrespondenzbeziehungen definiert - existierende *konkrete* Beziehungen zwischen Dokumenten bzw. Dokumentenmodellen. Ein Regelsatz für die Dokumentenebene wird *jeweils* für ein *bestimmtes* Paar von Dokumententypen modelliert und bedient sich insbesondere der Ausdrucksmittel für komplexe Graphmuster. Im Gegensatz dazu gibt es nur *einen* Regelsatz für die Dokumentenmodellebene, in dem insbesondere Ausdrucksmittel zur Ermittlung ähnlicher Elemente verwendet werden. Attributkorrespondenzen zwischen komponierten Typen sind in den Regeln ebenso definiert.

Die Ergebnisse dieser Arbeit verbessern das Vorgehen zur Erstellung von Integrationswerkzeugen. So kann ein Integrator jetzt zur Laufzeit konfiguriert werden. Des Weiteren sind einfach bedienbare Werkzeuge zur Regelmodellierung und Integrationsdurchführung entstanden, die leicht in bestehende Entwicklungsumgebungen integriert werden können. Durch diese Werkzeuge wird die Integration zusammen mit den neu eingeführten Reparaturaktionen realitätsnah in die Praxis umgesetzt.

Danksagung

Obwohl die Arbeit von mir allein verfasst ist, hatte ich vielfältige Unterstützung von Menschen aus meinem Umfeld, denen ich an dieser Stelle herzlich danken möchte.

Zunächst möchte ich Herrn Prof. Manfred Nagl für die interessante Aufgabenstellung, die Betreuung dieser Arbeit und die sonstige Unterstützung danken. Die Zeit war sehr schön und ausgesprochen lehrreich nicht zuletzt wegen der vielen Gespräche und seinem guten Vorbild. Außerdem gilt mein Dank Herrn Prof. Wilhelm Schäfer für die Übernahme des Zweitgutachtens zu dieser Arbeit sowie den Herren Prof. Stefan Kowalewski, Prof. Thomas Seidl und Prof. Peter Rossmanith für die Teilnahme als Vorsitzender bzw. Prüfer an meiner Doktorprüfung.

Mein besonderer Dank gilt Simon Becker, meinem Projektvorgänger, für die gute Einarbeitung und das inhaltliche Korrekturlesen dieser Arbeit.

Auch René Wörzberger möchte ich danken, dass er mir in der Endphase der Arbeit mit seinen Projekterfahrungen sehr helfen konnte. Seine Anregungen zur Darstellung der komplizierten Konzepte haben die Arbeit qualitativ verbessert.

Unterstützung hatte ich auch von meiner lieben Freundin Margit, die in kürzester Zeit die gesamte Arbeit gewissenhaft Korrektur gelesen und viele Formulierungsempfehlungen gegeben hat, ohne die der Ausdruck der Arbeit nicht so gelungen wäre. Sie war auch sonst stets für mich da, wofür ich ihr sehr dankbar bin.

Für die Vermittlung der Abläufe in verfahrenstechnischen Entwicklungsprozessen, die Anforderungsgenerierung und die Priorisierung von Funktionalitäten für Integratoren von Anwenderseite danke ich sehr Herrn Bernd Kokkelink, Produktmanager von Comos PT, sowie den Herren Walter Kattenbusch und Bernd Schneider der Firma Comos Industry Solutions GmbH für die technische Unterstützung.

Für die technische Umsetzung aber auch fachlichen Besprechungen möchte ich insbesondere den studentischen Hilfskräften und Diplomanden ganz herzlich danken. Am längsten - sogar länger als ich - war Tobias Campmann im Projekt dabei, der u. a. den Reparaturaktionengenerator realisiert hat. Es war mir immer ein besonderes Vergnügen, mit ihm zu arbeiten. Dann stießen Stefan Heukamp und Achim Drews dazu. Stefan realisierte die Dokumentenwiederverwendung, Achim baute den Dokumentenwrapper für Fließbilder in Comos. Am Ende waren Michael Brysch, Robert Schmidt und Konstantin Steinhauer für die Realisierung der Werkzeuge zuständig. Ohne ihre fleißige Unterstützung wäre die Arbeit nicht so gut verlaufen. Ich danke auch allen Beteiligten des Vorgängerprojekts, stellvertretend Marco Schmidt, der meine Konferenzbeiträge noch lange nach seiner produktiven Zeit Korrektur las.

Allen meinen Kollegen möchte ich für die schöne Zeit danken, die ich mit ihnen am Lehrstuhl hatte. Tapferer Mitstreiter aus dem Transferbereich war Thomas Heer. Ihm und Daniel Retkowitz zolle ich großen Respekt, weil sie es stets geschafft haben, auch in schwierigen Situationen die Ruhe zu bewahren. René Wörzberger und belustigte durch seinen Witz, wodurch jeder Tag am Lehrstuhl zu einem Original wurde. Mit Erhard Weinell, dem Linux-Guru, teilte ich das Büro. In gemeinsamen Lehrveranstaltungen bildeten wir das perfekte Team. Unsere Gespräche werde ich sehr ver-

missen. Mit Christof Mosler, meinem Lieblings-Immer-etwas-Neues-ausprobieren-Kollegen, habe ich während gemeinsamer Laufrunden und Besuchen im Polonia unzählige Businesspläne geschmiedet, die wir danach - vielleicht ist es auch besser so - nicht umgesetzt haben. Stephanie Mosler möchte ich an dieser Stelle dafür danken, dass sie diese Ideen immer richtig zu nehmen wusste. Thomas Haase hat mir viele Programmiertechniken beigebracht, Ulrike Ranger und Christian Fuss zeigten mir, wie man Lehrveranstaltungen organisiert.

Ibrahim Armac und Cem Mengi haben den Info-Cup ins Leben gerufen und durch den Aufbau einer sehr erfolgreichen Lehrstuhlmannschaft für ein besonders gutes Klima gesorgt, nicht zuletzt auch ihre Frauen Ebru und Azime. Zum Glück konnte ich die beiden Fußballgenies für die „andere Seite des Sports“ aufmuntern, dem Laufen. Zusammen mit René und den beiden neuen Kollegen aus Braunschweig, Jan Ringert und Martin Schindler, vertraten wir den Lehrstuhl beim Lousberglauf, für den wir gemeinsam trainiert hatten. Lautstarke Unterstützung an der Laufstrecke erhielten wir neben Thomas Heer auch von unseren jüngsten Kollegen Thomas Kurpick, Ingo Weisemöller und Arne Haber. Sie verkörpern ab nun den Lehrstuhl und vertreten ihn hoffentlich auch in folgenden Jahren bei sportlichen Wettkämpfen. Vielleicht sind dann auch die anderen „Neuen“ Claas Pinkernell, Christoph Herrmann und Christian Berger dabei. Zusammen mit Jan hatten sie die ehrenvolle Aufgabe, den Lehrstuhl von Dingen mit „historischem Wert“ zu befreien und ihn dafür mit frischem „niedersächsischem“ Wind zu bereichern. Prof. Bernhard Rumpe, dem neuen Lehrstuhlinhaber, verdanken wir fachlich gesehen neue Inspirationen und persönlich gesehen viel gute Laune, nicht zu vergessen einen neuen Kicker.

Konstanz in den Lehrstuhl brachten Marita Breuer, Galina Volkova, Angelika Fleck und Silke Cormann. Marita und Galina verkörperten das technische Wissen des Lehrstuhls der vergangenen Jahre, Angelika und Silke das organisatorische. Ihnen möchte ich sehr dafür danken, dass sie immer da waren, stets auf alle Fragen Antworten hatten und selbst bei kurzfristigen Angelegenheiten sehr hilfsbereit waren.

Danken möchte ich auch allen Mitarbeitern des Lehrstuhls vor meiner Zeit für die Entwicklung von Integratoren, PROGRES und TGG, deren Konzepte ich nutzen und erweitern durfte, stellvertretend den Profes. Andy Schürr und Albert Zündorf.

Neben den Personen aus meinem fachlichen Umfeld möchte ich all meinen Freunden danken, die mich beim Sport oder am Wochenende so schön auf andere Gedanken gebracht haben. Ohne diese Ablenkung wäre ich bestimmt nicht so motiviert wieder an die Arbeit gegangen. Schließlich danke ich meinem Freund Simon für seine humorvolle und gelassene Gesellschaft, die mir sehr geholfen hat.Von ihm habe ich die Lebensweisheiten „Et kütt wie et kütt“ und „Et hätt noch immer jot jejange“ gelernt, die mir eine völlig neue Lebensqualität gezeigt haben.

Aachen, 21. Juli 2009 *Anne-Thérèse Körtgen*

Inhaltsverzeichnis

Abbildungsverzeichnis ix

1 Einleitung 1
- 1.1 Konsistenzprobleme in Entwicklungsprozessen . . . 3
- 1.2 Verwandte Arbeiten . . . 6
  - 1.2.1 Einordnung . . . 7
  - 1.2.2 Tripelgraphgrammatiken . . . 10
  - 1.2.3 Konsistenzsicherung von Modellen . . . 12
  - 1.2.4 Arbeiten am Lehrstuhl für Informatik 3 . . . 18
  - 1.2.5 Integratorrahmenwerk aus dem SFB IMPROVE . . . 23
- 1.3 Nicht ausreichend unterstützte Eigenschaften . . . 36
  - 1.3.1 Reparaturaktionen in der Modellsynchronisation . . . 36
  - 1.3.2 Fehlende Eigenschaften der Transformationen . . . 38
  - 1.3.3 Herausforderungen im industriellen Einsatz . . . 39
- 1.4 Eigene Beiträge . . . 41
  - 1.4.1 Integrationsalgorithmus . . . 41
  - 1.4.2 Regelsprache . . . 43
  - 1.4.3 Regelerstellungsprozess . . . 45
  - 1.4.4 Benutzerinteraktion . . . 46
  - 1.4.5 Zusammenfassung . . . 47
- 1.5 Aufbau der Arbeit . . . 47

2 Szenario 49
- 2.1 Verfahrenstechnische Entwicklungsprozesse . . . 50
- 2.2 Konsistenzsicherung von Fließbildern . . . 53
  - 2.2.1 Inkrementelle Transformation . . . 54
  - 2.2.2 Reparaturen nach Weiterbearbeitung . . . 56
- 2.3 Regelinduktion und Korrespondenzanalyse . . . 59
  - 2.3.1 Ermittlung von Korrespondenzen bestehender Dokumente . . . 59
  - 2.3.2 Ermittlung von Typkorrespondenzen . . . 62
- 2.4 Weitere Szenarien . . . 65
- 2.5 Datenmodell der Fließbilder . . . 66

3 Modellierung der Konsistenzregeln 69
- 3.1 Regelvereinfachung . . . 69

3.2 Regelwiederverwendung . . . 73
3.3 Operationalisierung von Regeln . . . 78
3.4 Iterative und alternative Graphmuster . . . 82
3.4.1 Wiederholte Teilgraphen . . . 82
3.4.2 Alternative Teilgraphen . . . 87
3.5 Attributbehandlungen . . . 88
3.5.1 Inkrementvergleich . . . 89
3.5.2 Attributeinschränkung und -propagation . . . 92
3.6 Automatische Erstellung von Regeln . . . 95
3.6.1 Regelerstellung aus bestehenden Dokumenten . . . 96
3.6.2 Regelerstellung aus Dokumentenmodellen . . . 100
3.7 Verwandte Arbeiten . . . 111
3.7.1 Transformationssprachen . . . 112
3.7.2 Ansätze zum Schemamatching . . . 117
3.7.3 Generative Ansätze . . . 120

4 Transformationen zur Konsistenzwiederherstellung 125
4.1 Konsistente Regelanwendungsstelle . . . 127
4.2 Beschädigungs- und Reparaturarten . . . 129
4.3 Unbeschädigter Restteilgraph . . . 133
4.4 Reparaturaktionen . . . 134
4.4.1 Ersetzung durch alternative Kanten . . . 137
4.4.2 Ersetzung durch alternative Knoten . . . 138
4.4.3 Ersetzung durch alternativen Kontext . . . 141
4.4.4 Anwendung einer alternativen Regel . . . 143
4.4.5 Neue Integration . . . 144
4.4.6 Löschpropagation . . . 144
4.4.7 Attributpropagation . . . 146
4.4.8 Wiederherstellung der Attributbedingung . . . 146
4.4.9 Erzeugung fehlender Knoten und Kanten . . . 146
4.4.10 Wiederherstellung der NAC-Bedingung . . . 147
4.4.11 Wiederherstellung der Loop-Bedingung . . . 147
4.4.12 Verkleinerung der Regelanwendungsstelle . . . 148
4.5 Verwandte Arbeiten . . . 151

5 Algorithmen zur Regelausführung 153
5.1 Wiederherstellung der Korrespondenz . . . 153
5.1.1 Einbettung in den Integrationsprozess . . . 154
5.1.2 Reparaturaktionen generieren . . . 156
5.1.3 Reparaturoptionen finden . . . 156
5.1.4 Reparatur ausführen . . . 157
5.1.5 Prioritäten & Phasen . . . 160

5.2 Kontrollierte Regeltrigger 161
5.3 Ausführung neuer Ausdrucksmittel 168
5.3.1 Wiederholte und alternative Graphmuster 168
5.3.2 Attributgleichungen 185
5.4 Optimierte Korrespondenzanalyse 188
5.5 Verwandte Arbeiten 199

6 Realisierte Integrationsumgebung 201
6.1 Systemarchitektur 201
6.2 Erweiterungen am Integratorkern 204
6.2.1 Reparaturaktionen 206
6.2.2 Attributbehandlung 218
6.2.3 Triggerausführung 220
6.2.4 Kontextabhängigkeiten 223
6.2.5 Korrespondenzanalyse 225
6.2.6 Layoutpropagation 228
6.3 Management von Integrationsmodellen 230
6.3.1 Dokumente und Abhängigkeiten 230
6.3.2 Dokumenttypen und Abhängigkeiten 233
6.3.3 Regelmanager 234
6.3.4 Integration in Entwicklungsumgebungen 237
6.4 Unterstützung bei der Integratorenerstellung 239
6.4.1 Konfigurationswerkzeug 241
6.4.2 Regeleditor 245
6.4.3 Regelgenerator 260
6.5 Benutzerinteraktion während der Integration 263

7 Schlussbemerkungen 267
7.1 Zusammenfassung 267
7.2 Ausblick 269

Literaturverzeichnis 273

Related Interesting Work from the SE Group, RWTH Aachen

Agile Model Based Software Engineering

Agility and modeling in the same project? This question was raised in [Rum04]: "Using an executable, yet abstract and multi-view modeling language for modeling, designing and programming still allows to use an agile development process." Modeling will be used in development projects much more, if the benefits become evident early, e.g with executable UML [Rum02] and tests [Rum03]. In [GKRS06], for example, we concentrate on the integration of models and ordinary programming code. In [Rum12] and [Rum11], the UML/P, a variant of the UML especially designed for programming, refactoring and evolution, is defined. The language workbench MontiCore [GKR+06] is used to realize the UML/P [Sch12]. Links to further research, e.g., include a general discussion of how to manage and evolve models [LRSS10], a precise definition for model composition as well as model languages [HKR+09] and refactoring in various modeling and programming languages [PR03]. In [FHR08] we describe a set of general requirements for model quality. Finally [KRV06] discusses the additional roles and activities necessary in a DSL-based software development project.

Generative Software Engineering

The UML/P language family [Rum12, Rum11] is a simplified and semantically sound derivate of the UML designed for product and test code generation. [Sch12] describes a flexible generator for the UML/P based on the MontiCore language workbench [KRV10, GKR+06]. In [KRV06], we discuss additional roles necessary in a model-based software development project. In [GKRS06] we discuss mechanisms to keep generated and handwritten code separated. In [Wei12] we show how this looks like and how to systematically derive a transformation language in concrete syntax. To understand the implications of executability for UML, we discuss needs and advantages of executable modeling with UML in agile projects in [Rum04], how to apply UML for testing in [Rum03] and the advantages and perils of using modeling languages for programming in [Rum02].

Unified Modeling Language (UML)

Many of our contributions build on UML/P described in the two books [Rum11] and [Rum12] are implemented in [Sch12]. Semantic variation points of the UML are discussed in [GR11]. We discuss formal semantics for UML [BHP+98] and describe UML semantics using the "System Model" [BCGR09a], [BCGR09b], [BCR07b] and [BCR07a]. Semantic variation points have, e.g., been applied to define class diagram semantics [CGR08]. A precisely defined semantics for variations is applied, when checking variants of class diagrams [MRR11c] and objects diagrams [MRR11d] or the consistency of both kinds of diagrams [MRR11e]. We also apply these concepts to activity diagrams (ADs) [MRR11b] which allows us to check for semantic differences of activity diagrams [MRR11a]. We also discuss how to ensure and identify model quality [FHR08], how models, views and the system under development correlate to each other [BGH+98] and how to use modeling in agile development projects [Rum04], [Rum02] The question how to adapt and extend the UML in discussed in [PFR02] on product line annotations for UML and to more general discussions and insights on how to use meta-modeling for defining and adapting the UML [EFLR99], [SRVK10].

Domain Specific Languages (DSLs)

Computer science is about languages. Domain Specific Languages (DSLs) are better to use, but need appropriate tooling. The MontiCore language workbench [GKR+06], [KRV10], [Kra10] describes an integrated abstract and concrete syntax format [KRV07b] for easy development. New languages and tools

can be defined in modular forms [KRV08, Völ11] and can, thus, easily be reused. [Wei12] presents a tool that allows to create transformation rules tailored to an underlying DSL. Variability in DSL definitions has been examined in [GR11]. A successful application has been carried out in the Air Traffic Management domain [ZPK+11]. Based on the concepts described above, meta modeling, model analyses and model evolution have been examined in [LRSS10] and [SRVK10]. DSL quality [FHR08], instructions for defining views [GHK+07], guidelines to define DSLs [KKP+09] and Eclipse-based tooling for DSLs [KRV07a] complete the collection.

Modeling Software Architecture & the MontiArc Tool

Distributed interactive systems communicate via messages on a bus, discrete event signals, streams of telephone or video data, method invocation, or data structures passed between software services. We use streams, statemachines and components [BR07] as well as expressive forms of composition and refinement [PR99] for semantics. Furthermore, we built a concrete tooling infrastructure called MontiArc [HRR12] for architecture design and extensions for states [RRW13]. MontiArc was extended to describe variability [HRR+11] using deltas [HRRS11] and evolution on deltas [HRRS12]. [GHK+07] and [GHK+08] close the gap between the requirements and the logical architecture and [GKPR08] extends it to model variants. Co-evolution of architecture is discussed in [MMR10] and a modeling technique to describe dynamic architectures is shown in [HRR98].

Compositionality & Modularity of Models

[HKR+09] motivates the basic mechanisms for modularity and compositionality for modeling. The mechanisms for distributed systems are shown in [BR07] and algebraically underpinned in [HKR+07]. Semantic and methodical aspects of model composition [KRV08] led to the language workbench MontiCore [KRV10] that can even develop modeling tools in a compositional form. A set of DSL design guidelines incorporates reuse through this form of composition [KKP+09]. [Völ11] examines the composition of context conditions respectively the underlying infrastructure of the symbol table. Modular editor generation is discussed in [KRV07a].

Semantics of Modeling Languages

The meaning of semantics and its principles like underspecification, language precision and detailedness is discussed in [HR04]. We defined a semantic domain called “System Model” by using mathematical theory. [RKB95, BHP+98] and [GKR96, KRB96]. An extended version especially suited for the UML is given in [BCGR09b] and in [BCGR09a] its rationale is discussed. [BCR07a, BCR07b] contain detailed versions that are applied on class diagrams in [CGR08]. [MRR11a, MRR11b] encode a part of the semantics to handle semantic differences of activity diagrams and [MRR11e] compares class and object diagrams with regard to their semantics. In [BR07], a simplified mathematical model for distributed systems based on black-box behaviors of components is defined. Meta-modeling semantics is discussed in [EFLR99]. [BGH+97] discusses potential modeling languages for the description of an exemplary object interaction, today called sequence diagram. [BGH+98] discusses the relationships between a system, a view and a complete model in the context of the UML. [GR11] and [CGR09] discuss general requirements for a framework to describe semantic and syntactic variations of a modeling language. We apply these on class and object diagrams in [MRR11e] as well as activity diagrams in [GRR10]. [Rum12] embodies the semantics in a variety of code and test case generation, refactoring and evolution techniques. [LRSS10] discusses evolution and related issues in greater detail.

Evolution & Transformation of Models

Models are the central artifact in model driven development, but as code they are not initially correct and need to be changed, evolved and maintained over time. Model transformation is therefore essential to effectively deal with models. Many concrete model transformation problems are discussed: evolution [LRSS10, MMR10, Rum04], refinement [PR99, KPR97, PR94], refactoring [Rum12, PR03], translating models from one language into another [MRR11c, Rum12] and systematic model transformation language development [Wei12]. [Rum04] describes how comprehensible sets of such transformations support software development, maintenance and [LRSS10] technologies for evolving models within a language and across languages and linking architecture descriptions to their implementation [MMR10]. Automaton refinement is discussed in [PR94, KPR97], refining pipe-and-filter architectures is explained in [PR99]. Refactorings of models are important for model driven engineering as discussed in [PR03, Rum12]. Translation between languages, e.g., from class diagrams into Alloy [MRR11c] allows for comparing class diagrams on a semantic level.

Variability & Software Product Lines (SPL)

Many products exist in various variants, for example cars or mobile phones, where one manufacturer develops several products with many similarities but also many variations. Variants are managed in a Software Product Line (SPL) that captures the commonalities as well as the differences. Feature diagrams describe variability in a top down fashion, e.g., in the automotive domain [GHK+08] using 150% models. Reducing overhead and associated costs is discussed in [GRJA12]. Delta modeling is a bottom up technique starting with a small, but complete base variant. Features are added (that sometimes also modify the core). A set of applicable deltas configures a system variant. We discuss the application of this technique to Delta-MontiArc [HRR+11, HRR+11] and to Delta-Simulink [HKM+13]. Deltas can not only describe spacial variability but also temporal variability which allows for using them for software product line evolution [HRRS12]. [HHK+13] describes an approach to systematically derive delta languages. We also apply variability to modeling languages in order to describe syntactic and semantic variation points, e.g., in UML for frameworks [PFR02]. And we specified a systematic way to define variants of modeling languages [CGR09] and applied this as a semantic language refinement on Statecharts in [GR11].

Cyber-Physical Systems (CPS)

Cyber-Physical Systems (CPS) [KRS12] are complex, distributed systems which control physical entities. Contributions for individual aspects range from requirements [GRJA12], complete product lines [HRRW12], the improvement of engineering for distributed automotive systems [HRR12] and autonomous driving [BR12a] to processes and tools to improve the development as well as the product itself [BBR07]. In the aviation domain, a modeling language for uncertainty and safety events was developed, which is of interest for the European airspace [ZPK+11]. A component and connector architecture description language suitable for the specific challenges in robotics is discussed in [RRW13]. Monitoring for smart and energy efficient buildings is developed as Energy Navigator toolset [KPR12, FPPR12, KLPR12].

State Based Modeling (Automata)

Today, many computer science theories are based on state machines in various forms including Petri nets or temporal logics. Software engineering is particularly interested in using state machines for modeling systems. Our contributions to state based modeling can currently be split into three parts: (1) understanding how to model object-oriented and distributed software using statemachines resp. Statecharts

[GKR96, BCR07b, BCGR09b, BCGR09a], (2) understanding the refinement [PR94, RK96, Rum96] and composition [GR95] of statemachines, and (3) applying statemachines for modeling systems. In [Rum96] constructive transformation rules for refining automata behavior are given and proven correct. This theory is applied to features in [KPR97]. Statemachines are embedded in the composition and behavioral specifications concepts of Focus [BR07]. We apply these techniques, e.g., in MontiArcAutomaton [THR+13] as well as in building management systems [FLP+11].

Robotics

Robotics can be considered a special field within Cyber-Physical Systems which is defined by an inherent heterogeneity of involved domains, relevant platforms, and challenges. The engineering of robotics applications requires composition and interaction of diverse distributed software modules. This usually leads to complex monolithic software solutions hardly reusable, maintainable, and comprehensible, which hampers broad propagation of robotics applications. The MontiArcAutomaton language [RRW12] extends ADL MontiArc and integrates various implemented behavior modeling languages using MontiCore [RRW13] that perfectly fits Robotic architectural modelling. The LightRocks [THR+13] framework allows robotics experts and laymen to model robotic assembly tasks.

Automotive, Autonomic Driving & Intelligent Driver Assistance

Introducing and connecting sophisticated driver assistance, infotainment and communication systems as well as advanced active and passive safety-systems result in complex embedded systems. As these feature-driven subsystems may be arbitrarily combined by the customer, a huge amount of distinct variants needs to be managed, developed and tested. A consistent requirements management that connects requirements with features in all phases of the development for the automotive domain is described in [GRJA12]. The conceptual gap between requirements and the logical architecture of a car is closed in [GHK+07, GHK+08]. [HKM+13] describes a tool for delta modeling for Simulink [HKM+13]. [HRRW12] discusses means to extract a well-defined Software Product Line from a set of copy and paste variants. Quality assurance, especially of safety-related functions, is a highly important task. In the Carolo project [BR12a, BR12b], we developed a rigorous test infrastructure for intelligent, sensor-based functions through fully-automatic simulation [BBR07]. This technique allows a dramatic speedup in development and evolution of autonomous car functionality, and thus, enables us to develop software in an agile way [BR12a]. [MMR10] gives an overview of the current state-of-the-art in development and evolution on a more general level by considering any kind of critical system that relies on architectural descriptions. As tooling infrastructure, the SSElab storage, versioning and management services [HKR12] are essential for many projects.

Energy Management

In the past years, it became more and more evident that saving energy and reducing CO2 emissions is an important challenge. Thus, energy management in buildings as well as in neighbourhoods becomes equally important to efficiently use the generated energy. Within several research projects, we developed methodologies and solutions for integrating heterogeneous systems at different scales. During the design phase, the Energy Navigators Active Functional Specification (AFS) [FPPR12, KPR12] is used for technical specification of building services already. We adapted the well-known concept of statemachines to be able to describe different states of a facility and to validate it against the monitored values [FLP+11]. We show how our data model, the constraint rules and the evaluation approach to compare sensor data can be applied [KLPR12].

Cloud Computing & Enterprise Information Systems

The paradigm of Cloud Computing is arising out of a convergence of existing technologies for web-based application and service architectures with high complexity, criticality and new application domains. It promises to enable new business models, to lower the barrier for web-based innovations and to increase the efficiency and cost-effectiveness of web development. Application classes like Cyber-Physical Systems [KRS12], Big Data, App and Service Ecosystems bring attention to aspects like responsiveness, privacy and open platforms. Regardless of the application domain, developers of such systems are in need for robust methods and efficient, easy-to-use languages and tools. We tackle these challenges by perusing a model-based, generative approach [PR13]. The core of this approach are different modeling languages that describe different aspects of a cloud-based system in a concise and technology-agnostic way. Software architecture and infrastructure models describe the system and its physical distribution on a large scale. We apply cloud technology for the services we develop, e.g., the SSELab [HKR12] and the Energy Navigator [FPPR12, KPR12] but also for our tool demonstrators and our own development platforms. New services, e.g.,c collecting data from temperature, cars etc. are easily developed.

References

[BBR07] Christian Basarke, Christian Berger, and Bernhard Rumpe. Software & Systems Engineering Process and Tools for the Development of Autonomous Driving Intelligence. *Journal of Aerospace Computing, Information, and Communication (JACIC)*, 4(12):1158–1174, October 2007.

[BCGR09a] Manfred Broy, Maria Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. Considerations and Rationale for a UML System Model. In Kevin Lano, editor, *UML 2 Semantics and Applications*, pages 43–61. John Wiley & Sons, 2009.

[BCGR09b] Manfred Broy, Maria Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. Definition of the UML System Model. In Kevin Lano, editor, *UML 2 Semantics and Applications*, pages 63–93. John Wiley & Sons, 2009.

[BCR07a] Manfred Broy, Maria Victoria Cengarle, and Bernhard Rumpe. Towards a System Model for UML. Part 2: The Control Model. Technical Report TUM-I0710, TU Munich, February 2007.

[BCR07b] Manfred Broy, Maria Victoria Cengarle, and Bernhard Rumpe. Towards a System Model for UML. Part 3: The State Machine Model. Technical Report TUM-I0711, TU Munich, February 2007.

[BGH+97] Ruth Breu, Radu Grosu, Christoph Hofmann, Franz Huber, Ingolf Krüger, Bernhard Rumpe, Monika Schmidt, and Wolfgang Schwerin. Exemplary and Complete Object Interaction Descriptions. In H. Kilov, B. Rumpe, and I. Simmonds, editors, *Proceedings OOPSLA'97 Workshop on Object-oriented Behavioral Semantics*, TUM-I9737, TU Munich, 1997.

[BGH+98] Ruth Breu, Radu Grosu, Franz Huber, Bernhard Rumpe, and Wolfgang Schwerin. Systems, Views and Models of UML. In M. Schader and A. Korthaus, editors, *Proceedings of the Unified Modeling Language, Technical Aspects and Applications*. Physica Verlag, Heidelberg, 1998.

[BHP+98] Manfred Broy, Franz Huber, Barbara Paech, Bernhard Rumpe, and Katharina Spies. Software and System Modeling Based on a Unified Formal Semantics. In M. Broy and B. Rumpe, editors, *RTSE '97: Proceedings of the International Workshop on Requirements Targeting Software and Systems Engineering*, LNCS 1526, pages 43–68, Bernried, Germany, October 1998. Springer.

[BR07] Manfred Broy and Bernhard Rumpe. Modulare hierarchische Modellierung als Grundlage der Software- und Systementwicklung. *Informatik-Spektrum*, 30(1):3–18, Februar 2007.

[BR12a] Christian Berger and Bernhard Rumpe. Autonomous Driving - 5 Years after the Urban Challenge: The Anticipatory Vehicle as a Cyber-Physical System. In *Proceedings of the 10th Workshop on Automotive Software Engineering (ASE 2012)*, pages 789–798, Braunschweig, Germany, September 2012.

[BR12b] Christian Berger and Bernhard Rumpe. Engineering Autonomous Driving Software. In C. Rouff and M. Hinchey, editors, *Experience from the DARPA Urban Challenge*. Springer, 2012.

[CGR08] María Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. System Model Semantics of Class Diagrams. Informatik-Bericht 2008-05, CfG Fakultät, TU Braunschweig, 2008.

[CGR09] María Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. Variability within Modeling Language Definitions. In *Model Driven Engineering Languages and Systems. Proceedings of MODELS 2009*, LNCS 5795, pages 670–684, Denver, Colorado, USA, October 2009.

[EFLR99] Andy Evans, Robert France, Kevin Lano, and Bernhard Rumpe. Meta-Modelling Semantics of UML. In H. Kilov, B. Rumpe, and I. Simmonds, editors, *Behavioral Specifications of Businesses and Systems*. Kluver Academic Publisher, 1999.

[FHR08] Florian Fieber, Michaela Huhn, and Bernhard Rumpe. Modellqualität als Indikator für Softwarequalität: eine Taxonomie. *Informatik-Spektrum*, 31(5):408–424, Oktober 2008.

[FLP+11] Norbert Fisch, Markus Look, Claas Pinkernell, Stefan Plesser, and Bernhard Rumpe. State-Based Modeling of Buildings and Facilities. In *Proceedings of the 11th International Conference for Enhanced Building Operations (ICEBO' 11)*, New York City, USA, October 2011.

[FPPR12] Norbert Fisch, Claas Pinkernell, Stefan Plesser, and Bernhard Rumpe. The Energy Navigator - A Web-Platform for Performance Design and Management. In *Proceedings of the 7th International Conference on Energy Efficiency in Commercial Buildings (IEECB)*, Frankfurt a. M., Germany, April 2012.

[GHK+07] Hans Grönniger, Jochen Hartmann, Holger Krahn, Stefan Kriebel, and Bernhard Rumpe. View-based Modeling of Function Nets. In *Proceedings of the Object-oriented Modelling of Embedded Real-Time Systems (OMER4) Workshop*, Paderborn, Germany, October 2007.

[GHK+08] Hans Grönniger, Jochen Hartmann, Holger Krahn, Stefan Kriebel, Lutz Rothhardt, and Bernhard Rumpe. Modelling Automotive Function Nets with Views for Features, Variants, and Modes. In *Proceedings of 4th European Congress ERTS - Embedded Real Time Software*, Toulouse, 2008.

[GKPR08] Hans Grönniger, Holger Krahn, Claas Pinkernell, and Bernhard Rumpe. Modeling Variants of Automotive Systems using Views. In *Modellbasierte Entwicklung von eingebetteten Fahrzeugfunktionen (MBEFF)*, Informatik Bericht 2008-01, pages 76–89, CFG Fakultät, TU Braunschweig, March 2008.

[GKR96] Radu Grosu, Cornel Klein, and Bernhard Rumpe. Enhancing the SysLab System Model with State. Technical Report TUM-I9631, TUM, Munich, Germany, 1996.

[GKR+06] Hans Grönniger, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. MontiCore 1.0 - Ein Framework zur Erstellung und Verarbeitung domänspezifischer Sprachen. Technical Report 2006-04, CfG Fakultät, TU Braunschweig, August 2006.

[GKRS06] Hans Grönniger, Holger Krahn, Bernhard Rumpe, and Martin Schindler. Integration von Modellen in einen codebasierten Softwareentwicklungsprozess. In *Proceedings der Modellierung 2006*, Lecture Notes in Informatics LNI P-82, Innsbruck, März 2006. GI-Edition.

[GR95] Radu Grosu and Bernhard Rumpe. Concurrent Timed Port Automata. Technical Report TUM-I9533, TUM, Munich, Germany, 1995.

[GR11] Hans Grönniger and Bernhard Rumpe. Modeling Language Variability. In *Workshop on Modeling, Development and Verification of Adaptive Systems. 16th Monterey Workshop*, LNCS 6662, pages 17–32, Redmond, Microsoft Research, 2011. Springer.

[GRJA12] Tim Gülke, Bernhard Rumpe, Martin Jansen, and Joachim Axmann. High-Level Requirements Management and Complexity Costs in Automotive Development Projects: A Problem Statement. In *Requirements Engineering: Foundation for Software Quality. 18th International Working Conference, Proceedings, REFSQ 2012*, Essen, Germany, March 2012.

[GRR10] Hans Grönniger, Dirk Reiß, and Bernhard Rumpe. Towards a Semantics of Activity Diagrams with Semantic Variation Points. In *Model Driven Engineering Languages and Systems, Proceedings of MODELS*, LNCS 6394, Oslo, Norway, 2010. Springer.

[HHK+13] Arne Haber, Katrin Hölldobler, Carsten Kolassa, Markus Look, Klaus Müller, Bernhard Rumpe, and Ina Schaefer. Engineering Delta Modeling Languages. In *Proceedings of the 17th International Software Product Line Conference (SPLC), Tokyo*, pages 22–31. ACM, September 2013.

[HKM+13] Arne Haber, Carsten Kolassa, Peter Manhart, Pedram Mir Seyed Nazari, Bernhard Rumpe, and Ina Schaefer. First-Class Variability Modeling in Matlab / Simulink. In *Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems*, pages 11–18, New York, NY, USA, 2013. ACM.

[HKR+07] Christoph Herrmann, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. An Algebraic View on the Semantics of Model Composition. In D. H. Akehurst, R. Vogel, and R. F. Paige, editors, *Proceedings of the Third European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA 2007), Haifa, Israel*, pages 99–113. Springer, 2007.

[HKR+09] Christoph Herrmann, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. Scaling-Up Model-Based-Development for Large Heterogeneous Systems with Compositional Modeling. In H. Arabnia and H. Reza, editors, *Proceedings of the 2009 International Conference on Software Engineeering in Research and Practice*, Las Vegas, Nevada, USA, 2009.

[HKR12] Christoph Herrmann, Thomas Kurpick, and Bernhard Rumpe. SSELab: A Plug-In-Based Framework for Web-Based Project Portals. In *Proceedings of the 2nd International Workshop on Developing Tools as Plug-Ins (TOPI) at ICSE 2012*, pages 61–66, Zurich, Switzerland, June 2012. IEEE.

[HR04] David Harel and Bernhard Rumpe. Meaningful Modeling: What's the Semantics of "Semantics"? *IEEE Computer*, 37(10):64–72, Oct 2004.

[HRR98] Franz Huber, Andreas Rausch, and Bernhard Rumpe. Modeling Dynamic Component Interfaces. In Madhu Singh, Bertrand Meyer, Joseph Gil, and Richard Mitchell, editors, *TOOLS 26, Technology of Object-Oriented Languages and Systems*. IEEE Computer Society, 1998.

[HRR+11] Arne Haber, Holger Rendel, Bernhard Rumpe, Ina Schaefer, and Frank van der Linden. Hierarchical Variability Modeling for Software Architectures. In *Proceedings of International Software Product Lines Conference (SPLC 2011)*. IEEE Computer Society, August 2011.

[HRR12] Arne Haber, Jan Oliver Ringert, and Bernhard Rumpe. MontiArc - Architectural Modeling of Interactive Distributed and Cyber-Physical Systems. Technical Report AIB-2012-03, RWTH Aachen, February 2012.

[HRRS11] Arne Haber, Holger Rendel, Bernhard Rumpe, and Ina Schaefer. Delta Modeling for Software Architectures. *Tagungsband des Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme VII, fortiss GmbH*, February 2011.

[HRRS12] Arne Haber, Holger Rendel, Bernhard Rumpe, and Ina Schaefer. Evolving Delta-oriented Software Product Line Architectures. In *Large-Scale Complex IT Systems. Development, Operation and Management, 17th Monterey Workshop 2012*, LNCS 7539, pages 183–208, Oxford, UK, March 2012. Springer.

[HRRW12] Christian Hopp, Holger Rendel, Bernhard Rumpe, and Fabian Wolf. Einführung eines Produktlinienansatzes in die automotive Softwareentwicklung am Beispiel von Steuergerätesoftware. In *Software Engineering 2012: Fachtagung des GI-Fachbereichs Softwaretechnik in Berlin*, Lecture Notes in Informatics LNI 198, pages 181–192, 27. Februar - 2. März 2012.

[KKP+09] Gabor Karsai, Holger Krahn, Claas Pinkernell, Bernhard Rumpe, Martin Schindler, and Steven Völkel. Design Guidelines for Domain Specific Languages. In *Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM'09), Sprinkle, J., Gray, J., Rossi, M., Tolvanen, J.-P., (eds.)*, Techreport B-108, Helsinki School of Economics, Orlando, Florida, USA, October 2009.

[KLPR12] Thomas Kurpick, Markus Look, Claas Pinkernell, and Bernhard Rumpe. Modeling Cyber-Physical Systems: Model-Driven Specification of Energy Efficient Buildings. In *Proceedings of the Modelling of the Physical World Workshop MOTPW'12, Innsbruck, October 2012*, pages 2:1–2:6. ACM Digital Library, October 2012.

[KPR97] Cornel Klein, Christian Prehofer, and Bernhard Rumpe. Feature Specification and Refinement with State Transition Diagrams. In *Fourth IEEE Workshop on Feature Interactions in Telecommunications Networks and Distributed Systems*. P. Dini, IOS-Press, 1997.

[KPR12] Thomas Kurpick, Claas Pinkernell, and Bernhard Rumpe. Der Energie Navigator. In *Entwicklung und Evolution von Forschungssoftware. Tagungsband, Rolduc, 10.-11.11.2011*, Aachener Informatik-Berichte, Software Engineering Band 14. Shaker Verlag Aachen, 2012.

[Kra10] Holger Krahn. *MontiCore: Agile Entwicklung von domänenspezifischen Sprachen im Software-Engineering*. Aachener Informatik-Berichte, Software Engineering Band 1. Shaker Verlag, Aachen, Germany, 2010.

[KRB96] Cornel Klein, Bernhard Rumpe, and Manfred Broy. A stream-based mathematical model for distributed information processing systems - SysLab system model. In *Proceedings of the first International Workshop on Formal Methods for Open Object-based Distributed Systems*, pages 323–338. Chapmann & Hall, 1996.

[KRS12] Stefan Kowalewski, Bernhard Rumpe, and Andre Stollenwerk. Cyber-Physical Systems - eine Herausforderung für die Automatisierungstechnik? In *Proceedings of Automation 2012, VDI Berichte 2012*, pages 113–116. VDI Verlag, 2012.

[KRV06] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Roles in Software Development using Domain Specific Modelling Languages. In J. Gray, J.-P. Tolvanen, and J. Sprinkle, editors, *Proceedings of the 6th OOPSLA Workshop on Domain-Specific Modeling 2006 (DSM'06), Portland, Oregon USA*, Technical Report TR-37, pages 150–158, Jyväskylä University, Finland, 2006.

[KRV07a] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Efficient Editor Generation for Compositional DSLs in Eclipse. In *Proceedings of the 7th OOPSLA Workshop on Domain-Specific Modeling (DSM' 07), Montreal, Quebec, Canada*, Technical Report TR-38, pages 8–10, Jyväskylä University, Finland, 2007.

[KRV07b] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Integrated Definition of Abstract and Concrete Syntax for Textual Languages. In G. Engels, B. Opdyke, D. C. Schmidt, and F. Weil, editors, *Proceedings of the ACM/IEEE 10th International Conference on Model Driven Engineering Languages and Systems (MODELS 2007), Nashville, TN, USA, October 2007*, LNCS 4735. Springer, 2007.

[KRV08] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Monticore: Modular Development of Textual Domain Specific Languages. In R. F. Paige and B. Meyer, editors, *Proceedings of the 46th International Conference Objects, Models, Components, Patterns (TOOLS-Europe), Zurich, Switzerland, 2008*, Lecture Notes in Business Information Processing LNBIP 11, pages 297–315. Springer, 2008.

[KRV10] Holger Krahn, Bernhard Rumpe, and Stefen Völkel. MontiCore: a Framework for Compositional Development of Domain Specific Languages. *International Journal on Software Tools for Technology Transfer (STTT)*, 12(5):353–372, September 2010.

[LRSS10] Tihamer Levendovszky, Bernhard Rumpe, Bernhard Schätz, and Jonathan Sprinkle. Model Evolution and Management. In *MBEERTS: Model-Based Engineering of Embedded Real-Time Systems, International Dagstuhl Workshop, Dagstuhl Castle, Germany*, LNCS 6100, pages 241–270. Springer, October 2010.

[MMR10] Tom Mens, Jeff Magee, and Bernhard Rumpe. Evolving Software Architecture Descriptions of Critical Systems. *IEEE Computer*, 43(5):42–48, May 2010.

[MRR11a] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. ADDiff: Semantic Differencing for Activity Diagrams. In *Proc. Euro. Soft. Eng. Conf. and SIGSOFT Symp. on the Foundations of Soft. Eng. (ESEC/FSE'11)*, pages 179–189. ACM, 2011.

[MRR11b] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. An Operational Semantics for Activity Diagrams using SMV. Technical Report AIB-2011-07, RWTH Aachen University, Aachen, Germany, July 2011.

[MRR11c] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. CD2Alloy: Class Diagrams Analysis Using Alloy Revisited. In *Model Driven Engineering Languages and Systems (MODELS 2011), Wellington, New Zealand*, LNCS 6981, pages 592–607, 2011.

[MRR11d] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. Modal Object Diagrams. In *Proc. 25th Euro. Conf. on Object Oriented Programming (ECOOP'11)*, LNCS 6813, pages 281–305. Springer, 2011.

[MRR11e] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. Semantically Configurable Consistency Analysis for Class and Object Diagrams. In *Model Driven Engineering Languages and Systems (MODELS 2011), Wellington, New Zealand*, LNCS 6981, pages 153–167. Springer, 2011.

[PFR02] Wolfgang Pree, Marcus Fontoura, and Bernhard Rumpe. Product Line Annotations with UML-F. In G. J. Chastek, editor, *Software Product Lines - Second International Conference, SPLC 2*, LNCS 2379, pages 188–197, San Diego, 2002. Springer.

[PR94] Barbara Paech and Bernhard Rumpe. A new Concept of Refinement used for Behaviour Modelling with Automata. In M. Naftalin, T. Denvir, and M. Bertran, editors, *FME'94: Industrial Benefit of Formal Methods*, LNCS 873. Springer, October 1994.

[PR99] Jan Philipps and Bernhard Rumpe. Refinement of Pipe-and-Filter Architectures. In J. Davies J. M. Wing, J. Woodcock, editor, *FM'99 - Formal Methods, Proceedings of the World Congress on Formal Methods in the Development of Computing System*, LNCS 1708, pages 96–115. Springer, 1999.

[PR03] Jan Philipps and Bernhard Rumpe. Refactoring of Programs and Specifications. In H. Kilov and K. Baclawski, editors, *Practical foundations of business and system specifications*, pages 281–297. Kluwer Academic Publishers, 2003.

[PR13] Antonio Navarro Perez and Bernhard Rumpe. Modeling Cloud Architectures as Interactive Systems. In I. Ober, A. S. Gokhale, J. H. Hill, J. Bruel, M. Felderer, D. Lugato, and A. Dabholka, editors, *Proc. of the 2nd International Workshop on Model-Driven Engineering for High Performance and Cloud Computing. Co-located with MODELS 2013, Miami*, Sun SITE Central Europe Workshop Proceedings CEUR 1118, pages 15–24. CEUR-WS.org, 2013.

[RK96] Bernhard Rumpe and Cornel Klein. Automata Describing Object Behavior. In H. Kilov and W. Harvey, editors, *Specification of Behavioral Semantics in Object-Oriented Information Modeling*, pages 265–286. Kluwer Academic Publishers, 1996.

[RKB95] Bernhard Rumpe, Cornel Klein, and Manfred Broy. Ein strombasiertes mathematisches Modell verteilter informationsverarbeitender Systeme - Syslab-Systemmodell. Technical Report TUM-I9510, Technische Universität München, 1995.

[RRW12] Jan Oliver Ringert, Bernhard Rumpe, and Andreas Wortmann. A Requirements Modeling Language for the Component Behavior of Cyber Physical Robotics Systems. In N. Seyff and A. Koziolek, editors, *Modelling and Quality in Requirements Engineering: Essays Dedicated to Martin Glinz on the Occasion of His 60th Birthday*. Monsenstein und Vannerdat, Münster, 2012.

[RRW13] Jan Oliver Ringert, Bernhard Rumpe, and Andreas Wortmann. MontiArcAutomaton: Modeling Architecture and Behavior of Robotic Systems. In *Workshops and Tutorials Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA), May 6-10, 2013, Karlsruhe, Germany*, pages 10–12, 2013.

[Rum96] Bernhard Rumpe. *Formale Methodik des Entwurfs verteilter objektorientierter Systeme*. Herbert Utz Verlag Wissenschaft, ISBN 3-89675-149-2, 1996.

[Rum02] Bernhard Rumpe. Executable Modeling with UML - A Vision or a Nightmare? In T. Clark and J. Warmer, editors, *Issues & Trends of Information Technology Management in Contemporary Associations, Seattle*, pages 697–701. Idea Group Publishing, Hershey, London, 2002.

[Rum03] Bernhard Rumpe. Model-Based Testing of Object-Oriented Systems. In F. de Boer, M. Bonsangue, S. Graf, W.-P. de Roever, editor, *Formal Methods for Components and Objects*, LNCS 2852, pages 380–402. Springer, November 2003.

[Rum04] Bernhard Rumpe. Agile Modeling with the UML. In M. Wirsing, A. Knapp, and S. Balsamo, editors, *Radical Innovations of Software and Systems Engineering in the Future. 9th International Workshop, RISSEF 2002. Venice, Italy, October 2002*, LNCS 2941. Springer, October 2004.

[Rum11] Bernhard Rumpe. *Modellierung mit UML*. Springer, second edition, September 2011.

[Rum12] Bernhard Rumpe. *Agile Modellierung mit UML: Codegenerierung, Testfälle, Refactoring*. Springer, second edition, Juni 2012.

[Sch12] Martin Schindler. *Eine Werkzeuginfrastruktur zur agilen Entwicklung mit der UML/P*. Aachener Informatik-Berichte, Software Engineering Band 11. Shaker Verlag, Aachen, Germany, 2012.

[SRVK10] Jonathan Sprinkle, Bernhard Rumpe, Hans Vangheluwe, and Gabor Karsai. Metamodelling: State of the Art and Research Challenges. In *MBEERTS: Model-Based Engineering of Embedded Real-Time Systems, International Dagstuhl Workshop, Dagstuhl Castle, Germany*, LNCS 6100, pages 57–76, October 2010.

[THR+13] Ulrike Thomas, Gerd Hirzinger, Bernhard Rumpe, Christoph Schulze, and Andreas Wortmann. A New Skill Based Robot Programming Language Using UML/P Statecharts. In *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 461–466, Karlsruhe, Germany, May 2013. IEEE.

[Völ11] Steven Völkel. *Kompositionale Entwicklung domänenspezifischer Sprachen*. Aachener Informatik-Berichte, Software Engineering Band 9. Shaker Verlag, Aachen, Germany, 2011.

[Wei12] Ingo Weisemöller. *Generierung domänenspezifischer Transformationssprachen*. Aachener Informatik-Berichte, Software Engineering Band 12. Shaker Verlag, Aachen, Germany, 2012.

[ZPK+11] Massimiliano Zanin, David Perez, Dimitrios S Kolovos, Richard F Paige, Kumardev Chatterjee, Andreas Horst, and Bernhard Rumpe. On Demand Data Analysis and Filtering for Inaccurate Flight Trajectories. In D. Schaefer, editor, *Proceedings of the SESAR Innovation Days*. EUROCONTROL, November 2011.