

TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Informatik XIX

Federated Enterprise Architecture Model Management

—

Conceptual Foundations, Collaborative Model Integration, and Software Support

Sascha Roth

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Georg Carle

Prüfer der Dissertation:

1. Univ.-Prof. Dr. Florian Matthes
2. Univ.-Prof. Dr. Ulrich Frank, Universität Duisburg-Essen

Die Dissertation wurde am 01.07.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 28.10.2014 angenommen.

Abstract

Enterprise modeling communities such as business process management, strategic IT management, or project portfolio management are supported by highly specialized models representing domain knowledge in a formal manner. Although knowledge of modeling communities is maintained locally, subsets thereof are of high interest for the entire organization, e.g. to perform holistic analyses. This makes synchronization of changes essential such that modeling communities share up-to-date information with each other. At the same time, each modeling community wants to keep control over its model and metamodel. Enterprise architects act as mediators between these different communities to provide their stakeholders with up-to-date information. Enterprise Architecture (EA) management seeks to foster the mutual alignment of business and IT. This requires both, a holistic management and a sound decision base, which is commonly captured in an EA model. Enterprise architects retrieve information from diverse autonomous modeling communities to maintain a central, coherent, and consistent EA model. Our empirical findings confirm that enterprise architects consider the maintenance of an EA model as time-consuming, cost-intensive, and error-prone and that they procure EA information primarily from existing model repositories. A particular challenge is to continuously integrate model changes from specialized and autonomous modeling communities into a holistic and consistent EA model. Since each modeling community remains autonomous, conflicts between models as well as metamodels can arise when independent communities concurrently alter models. In this process, most of these conflicts cannot be resolved automatically or by a single user, but require cross-community collaboration.

In this thesis, we assume that the EA model and models of other modeling communities form a federation. We illuminate typical characteristics of such a federation and of Federated EA Model Management such as number of elements or direction of information flows and provide delimitations to existing approaches from federated databases, software merging, and version control systems. Based on these characteristics we develop MODELGLUE, a socio-technical solution facilitating a software-supported process to continuously integrate object-oriented models of decentralized special purpose repositories into a central EA model. MODELGLUE includes model integration mechanisms and tolerates model conflicts to offer the necessary degree of freedom for a federated management of models. We detail concepts for conflict detection in different models, for user awareness, and for adaptable resolution strategies. MODELGLUE empowers users to resolve model and metamodel conflicts in a collaborative fashion by providing tasks aimed at resolving conflicts and an interactive conflict management dashboard. This way, MODELGLUE provides means to increase consistency in all models that participate in the federation. We describe an implementation of MODELGLUE based on a state-of-the-art Enterprise 2.0 solution, summarize user feedback from two case studies in practice, and report on further insights from an interview series.

Zusammenfassung

Modellierungsgemeinschaften, wie z. B. das Business-Prozess-Management, das strategische IT-Management oder das Projekt-Portfolio-Management, werden von hochspeziellen Modellen unterstützt, die Domänenwissen formal repräsentieren. Das Wissen dieser Modellierungsgemeinschaften wird meist lokal verwaltet; gewisse Teilaspekte sind jedoch auch für die Gesamtorganisation relevant, z. B. im Rahmen von ganzheitlichen Analysen. Die Synchronisation von Modelländerungen ist essentiell, um aktuelle Informationen der Modellierungsgemeinschaften miteinander zu teilen. Dabei wollen die Modellierungsgemeinschaften die uneingeschränkte Kontrolle über ihr Modell und Metamodell behalten. Enterprise Architekten agieren als Mediatoren zwischen verschiedenen Modellierungsgemeinschaften, um ihre Interessensgruppen mit aktuellen Informationen zu versorgen. Im Rahmen von Enterprise Architecture (EA) Management Initiativen versuchen sie das Business und die IT aneinander auszurichten. Das verlangt ein ganzheitliches Management und eine fundierte Entscheidungsbasis, die in einem EA Modell verwaltet wird. Um dieses zentrale EA Modell zu pflegen, beziehen sie relevante Informationen aus verschiedenen autonomen Modellierungsgemeinschaften. Unsere empirischen Ergebnisse untermauern, dass Enterprise Architekten die Pflege eines EA Modells für zeitaufwändig, kostenintensiv und fehleranfällig erachten. Besonders auffallend ist, dass sie EA Informationen primär von existierenden Modellierungssystemen beziehen. Die kontinuierliche Integration von Modelländerungen autonomer Modellierungsgemeinschaften in ein ganzheitliches und konsistentes EA Modell stellt eine besondere Herausforderung dar, weil die gleichzeitige Modellierung in den Modellierungsgemeinschaften zu Konflikten zwischen den involvierten Modellen und Metamodellen führt. Die meisten dieser Konflikte können weder automatisch noch von einzelnen Nutzern aufgelöst werden, so dass eine gemeinschaftsübergreifende Kollaboration meist unabdingbar ist.

In dieser Arbeit gehen wir davon aus, dass das EA Modell und die Modelle der Modellierungsgemeinschaften eine Föderation bilden. Wir erläutern typische Charakteristiken einer solchen Föderation und Föderierten-EA-Modell-Managements, wie z.B. die Anzahl der Elemente oder die Richtung des Informationsflusses, und nehmen eine Abgrenzung gegenüber existierenden Ansätzen aus föderierten Datenbanken, Software Merging und Versionskontrollsystemen vor. Basierend auf diesen Charakteristiken entwerfen wir MODELGLUE, eine sozio-technische Lösung die einen softwaregestützten Prozess zur kontinuierlichen Integration von objekt-orientierten Modellen dezentraler Spezialsysteme in ein zentrales EA Modell unterstützt. Die Integrationsmechanismen von MODELGLUE tolerieren Modellkonflikte und realisieren so einen notwendigen Grad an Freiheit für das föderierte Management von Modellen. Wir detaillieren Konzepte für die Konflikterkennung in verschiedenen Modellen, für ein Konfliktbewusstsein bei Nutzern und für adaptierbare Lösungsstrategien. Unser Ansatz nutzt Tasks die zur Konfliktauflösung gedacht sind und ein interaktives Konflikt-Management-Dashboard, das Nutzern ermöglicht, Modell- und Metamodellkonflikte auf kollaborative Art zu lösen. MODELGLUE ist ein Hilfsmittel, um die Konsistenz in allen Modellen, die der Föderation angehören, zu erhöhen. Wir beschreiben eine Implementierung von MODELGLUE auf Basis einer existierenden Enterprise 2.0 Lösung, fassen Feedback von Nutzern in zwei Fallstudien zusammen und berichten über weitere Erkenntnisse einer Interviewserie.

Acknowledgment

As this journey comes to an end, I want to use this page to express my gratitude to the individuals that supported me throughout my research which is reported in this thesis.

For making all of this possible and for taking me under his wings, I would like to express my special appreciation and thanks to my doctoral father and first supervisor Prof. Dr. Florian Matthes. Thank you for all the inspiring discussions that finally led to this thesis. My thanks go also to my second supervisor Prof. Dr. Ulrich Frank. Thank you for the valuable feedback that focused and strengthened the contributions of this thesis.

In many settings I experienced that an individual is only as strong as the whole team. That being said, it was an honor to work with such an innovative and inspiring team. In alphabetical order my thanks go to my former and current colleagues: Dr. Sabine Buckl, Dr. Thomas Büchner, Matheus Hauder, Moritz Mack, Ivan Monahov, Dr. Christian Neubert, Thomas Reschenhofer, Alexander W. Schneider, Dr. Christopher Schulz, Dr. Christian M. Schweda, Klym Shumaiev, Dr. Alexander Steinhoff, and Marin Zec.

During the years, I had the opportunity to guide several students through their research. Though, sometimes it was not only the students learning. In alphabetical order, my thanks go to: Pouya Aleatrati Khosroshahi, Kamil Fabisiewicz, Steffen Fuhrmann, Sebastian Grunow, Tobias Höfler, Tong Lin, Mariana Mykhashchuk, Nevzat Orhan, and Michael Schaub. In several ways, student assistants that worked at our chair contributed to the success my work. To some of them, I would like to say ‘thank you’: Max Fiedler, Ermal Guni, Björn Kirschner, Felix Michel, Dominik Münch, Simon Pigat, Tobias Schrade, and Alexej Utz.

On several occasions throughout my research, I did not only work with colleagues and students but also with other researchers and industry partners. My thanks go to Jara Pascual and Jürgen Freygang from Adidas, Dr. Andreas Gehlert from the Federal Ministry of the Interior, Martin Baumann from Kassenärztliche Vereinigung Bayerns (KVB), Karsten Hahn and Erich Wohnig from HUK Coburg, Markus Buschle and Mathias Ekstedt from KTH Stockholm as well as Matthias Farwick and Prof. Dr. Ruth Breu from University of Innsbruck.

My final thanks go to my family, who spent most of the moral support. I am very grateful that you believe in me. For your constant and unconditional love and support, my heartwarming thanks go to my mother Gabriele Roth, my father Jürgen W. Roth, and my sister Melanie Roth.

Munich, July, 1, 2014

Sascha Roth

1	Introduction	1
1.1	Problem Statement	2
1.2	Research Questions	6
1.3	Research Design	8
1.4	Contributions	11
1.5	Structure of the Thesis	13
2	Foundations	15
2.1	EA and EA Management	16
2.1.1	A Structural Perspective on an Enterprise Architecture	16
2.1.2	Conceptual Framework of the ISO 42010 Standard	19
2.1.3	EA Management Interrelated with other Management Disciplines	21
2.1.4	EA Management is a Continuous Process	23
2.1.5	Summary of EA and EA Management	26
2.2	Modeling and EA Modeling	27
2.2.1	Fundamental Properties of a Model	27
2.2.2	Meta-levels of Models	28
2.3	Model Evolution and Merging	30
2.3.1	Two-way, Three-way, and N-way Merging	31
2.3.2	Textual, Syntactic, Semantic, Structural Merging	33
2.3.3	State-based, Change-based, and Operation-based Merging	34
2.3.4	Static Identity-based, Signature-based, and Similarity-based Matching	35
2.3.5	Tentative Merge Results	36
2.3.6	Conflict Detection	36
2.3.7	Conflict Avoidance, Tolerance, and Resolution	37
2.3.8	Collaboration	38
2.3.9	Persistent Versions	38
2.3.10	Visualizations	39
2.3.11	Technology Stack	40
2.3.12	Summary of Model Evolution & Merging	40
2.4	Federated Database Systems	41

2.4.1	Interacting Schemas in Federated Database Systems	42
2.4.2	Access Rights	44
2.4.3	Exchange of Schemas	45
2.4.4	Negotiation and Transactions	45
2.4.5	Replication of Information and Provenance	45
2.4.6	Integrity	46
2.4.7	Evolution Process of Federated Database Systems	46
2.4.8	Summary of Federated Database Systems	47
2.5	Summary	48
3	State-of-the-Art in EA Model Maintenance	49
3.1	Research Topics in EA Model Maintenance	50
3.2	Organizational Aspects	54
3.2.1	Incentives	54
3.2.2	Roles	55
3.2.3	Change Events	56
3.3	Process Descriptions	56
3.3.1	Conflict Resolution	58
3.3.2	Collaboration	59
3.4	Tools and Techniques	59
3.4.1	Collection	59
3.4.2	Experience Reports	59
3.4.3	Quality Assurance	60
3.4.4	Conflict Management	60
3.4.5	Integration	60
3.5	Summary	61
4	Requirements Analysis	63
4.1	Conceptual Overview of Federated EA Model Management	63
4.1.1	Roles within Federated EA Model Management	68
4.1.2	Repositories of Modeling Communities	73
4.1.3	Models within a Federated EA Model Environment	78
4.1.4	Consistency within a Federated EA Model Environment	83
4.2	Use Case Analysis of Federated EA Model Management	85
4.2.1	A Template for the Structured Presentation of Use Cases	85
4.2.2	Integrate Information Source	88
4.2.3	Configure Mapping	91
4.2.4	Synchronize Information Source	95
4.2.5	Import Information	97
4.2.6	View Differences	101
4.2.7	Merge Models	102
4.2.8	Resolve Conflict	105
4.2.9	Resolve Conflicts Collaboratively	107
4.2.10	Configure Conflict Resolution	109
4.2.11	Assign Task	112
4.2.12	View Conflicts	114
4.2.13	Additional Use Cases within Federated EA Model Management . . .	116

4.3	Requirements for a Federated EA Model Management Solution	118
4.3.1	Process Requirements	119
4.3.2	Collaboration Requirements	120
4.3.3	Modeling Requirements	122
4.3.4	Usability Requirements	126
4.3.5	Technology Requirements	127
4.4	Summary	128
5	Federated EA Model Management Design	131
5.1	System Design	131
5.1.1	Overview of MODELGLUE	132
5.1.2	A Metamodel for Evolutionary EA Modeling in a Federated Environment	133
5.2	Process Design	142
5.2.1	A Federated Model Environment as Branches of a Federal Model . .	144
5.2.2	Integrate Information Sources	146
5.2.3	Importing Information	157
5.2.4	Model Differencing	162
5.2.5	Merging Models and Metamodels	167
5.2.6	Manual Detection of Conflicts	179
5.2.7	Resolve Conflicts	185
5.2.8	Adaptable Conflict Resolution Strategies	189
5.3	Interactive Visual Support	198
5.3.1	Visualizing Metamodel and Model Differences	198
5.3.2	An Interactive and Collaborative Conflict Management Dashboard .	207
5.4	Summary	213
6	Software Support for Federated EA Model Management	215
6.1	Architecture of MODELGLUE	215
6.1.1	Configuration of Variability Points	216
6.2	Extending a Non-rigid Typed Repository with Tasks	218
6.2.1	Roles and Model Actions	218
6.2.2	Tasks	222
6.3	A Framework for Interactive Visualizations	225
6.3.1	Architecture	225
6.3.2	A Model for Interactive Visualizations	227
6.3.3	Towards Processing an Interactive Visualization	237
6.3.4	Visualization Process	240
6.4	Implementing a Real-time Conflict Management Dashboard	242
6.4.1	Layout and Layers	242
6.4.2	Conflict Tasks	243
6.4.3	Real-time Collaboration	244
6.5	Summary	248
7	Evaluation	249
7.1	Evaluation Design	249
7.2	Case Study in the German Insurance Sector	252
7.2.1	Initial Interviews	253

7.2.2	Context	254
7.2.3	Execution, Feedback, and Adaptations	259
7.2.4	Reflection of Results	267
7.3	Case Study in the German Health Industry	267
7.3.1	Initial Interviews	268
7.3.2	Context	268
7.3.3	Execution, Feedback, and Adaptations	270
7.3.4	Reflection of Results	279
7.4	Conclusion of the Case Studies	280
7.5	Interview Series	281
7.5.1	Status-quo in Industry	281
7.5.2	Benefits of Federated EA Model Management	281
7.5.3	Additional Process Steps	282
7.5.4	Degree of Automation	282
7.5.5	Model Changes and Conflict Resolution	282
7.5.6	Additional Factors	283
7.5.7	Reflection of the Interview Results	283
7.6	Summary	284
8	Conclusion	285
8.1	Summary	285
8.2	Critical Reflection	290
8.2.1	Validity of the Conclusions	290
8.2.2	Detection of Changes within Information Sources	290
8.2.3	Modeling Capabilities	292
8.2.4	Implementation Aspects	292
8.2.5	Non-functional Requirements	292
8.3	Further Research	292
8.3.1	Additional Case Studies in Real-World Setting	292
8.3.2	Business Cases for Federated EA Model Management	293
8.3.3	Standardization of Models and Domain Ontologies for EA Management	293
8.3.4	Usability Experiments	294
8.3.5	Administering Planned States of an EA Model	294
8.3.6	De-contextualizing the Visualizations	295
8.3.7	De-contextualizing Tasks for Adaptive Case Management	295
A	Import Schema	297
B	Mapping of the Conceptual Model to the Implementation	302
B.1	ModelElements and Roles	303
B.2	Changesets and Operations	304
	Bibliography	305
	List of Acronyms	335
	Index	339

List of Figures

1.1	Information exchange in a federated EA model environment as a socio-technical system of systems	4
1.2	Information systems research framework of Hevner et al. [HMP ⁺ 04]	9
1.3	Research questions, structure of the thesis, and related publications	12
2.1	Intersection of different areas of related work for the present thesis	15
2.2	Fundamental structure of a holistic view on an enterprise [BEG ⁺ 12]	17
2.3	An EA as cross-layer view of aggregated artifacts [WF06]	18
2.4	Conceptual framework of the ISO 42010 standard for architectural descriptions of software-intensive systems [ISO07b]	20
2.5	Integration of EA management with other management functions (based on [Wi07, p. 12])	21
2.6	EA management as an iterative management discipline (based on [RZM14, p. 12])	23
2.7	Relationship of model, metamodel, meta-metamodel, and meta-meta-metamodel	29
2.8	Merge of source code using the Unix diff and diff3 utilities	32
2.9	Component, FDBS, and interrelated schemas in the reference FDBS architecture of Sheth and Larson	44
2.10	Message exchange between two components in a federated architecture	46
3.1	A topic map of EA model maintenance research	51
4.1	Conceptual overview of Federated EA Model Management as a socio-technical system of systems	64
4.2	Stakeholders of a federated EA model environment: roles of the EA team and modeling communities	70
4.3	System usage and relevance as EA information sources (n=123) [FBH ⁺ 13]	74
4.4	Empirical results on information quality attributes of different information sources [FBH ⁺ 13]	75
4.5	Classification of different information sources within an enterprise	76

4.6	Semantic relationship between an EA model and specialized models	78
4.7	Entities and relationships of models of EAM, PPM, ITSM	81
4.8	Use Cases of Federated EA Model Management	86
4.9	Relationship of a common metamodel, import metamodel, export metamodel, import model and export model	89
4.10	Non-exhaustive list of examples for an executed mapping of attributes of an export model to an import model	94
4.11	An initial conceptual notion of tasks within Federated EA Model Management	114
5.1	Conceptual overview of MODELGLUE: EA model management in a federated environment	132
5.2	A language for Federated EA Model Management	134
5.3	States of a MODELELEMENT	138
5.4	States of a TASK	140
5.5	Linguistic and ontological instantiation of core concepts in our metamodel based on Kühne [Kü06]	142
5.6	Process model for integrating information sources in a federation	143
5.7	Topic map on the term integration and the perspectives taken in our approach	147
5.8	Overview of the sub-process integrate information source	149
5.9	Different types of communication within a federated EA model environment	151
5.10	A taxonomy of data quality problems according to Rahm and Do [RD00] .	153
5.11	Simplified view of the Ecore kernel based on [SBP ⁺ 09, ch. 5]	155
5.12	Illustrative mapping of a relational database to the target metamodel . . .	157
5.13	Importer: initiating a connection and querying for objects	158
5.14	Importer: updating an object	159
5.15	Relationship between chunk size and system load (schematic)	160
5.16	Scope of the differencing algorithm and the differencing visualization as an annotated Venn diagram	163
5.17	A metamodel for calculating differences between different models	163
5.18	Different evolution paths with the same resulting state	168
5.19	Operation-based tracking of modifications using CHANGESETS and CHANGES	172
5.20	Abstraction gap within an EA model	180
5.21	User interface of MODELGLUE: interactions with the worklist and tasks . .	182
5.22	User interface of MODELGLUE: configuring an abstraction gap resolution task	183
5.23	User interface of MODELGLUE: resolving an abstraction gap (annotations added to the screenshot as presented in [RHM13b])	184
5.24	Conflict resolution sub-process [RHM13b]	185
5.25	Learning mechanism as proposed by Schrade	188
5.26	Confirmation dialog developed by Schrade [Sc13] and as illustrated by Kirschner in [Ki14, p. 47]	189
5.27	Overview of conflict resolution strategies	190
5.28	Conflict resolution strategy widget	194
5.29	Filter to customize playlists in iTunes	195
5.30	Annotated screenshot of the dialog to specify merge rules developed by Kirschner [Ki14, p. 46])	196
5.31	Screenshot of the dialog to specify merge rules (minified)	197
5.32	Screenshot of the dialog to specify merge rules (expanded)	197

5.33	Four-layered interactive difference visualization for EA metamodels and models	199
5.34	Layer 1: differences of two metamodels (schemas) A and B with aggregated information of differences in respective models (data)	200
5.35	Layer 2: the instance overview with object details on mouse hover	202
5.36	Layer 3: graph representation of an 1st-neighborhood of an instance	203
5.37	Layer 4: three-way difference view of an instance (top) and other layers (bottom/background)	204
5.38	User interface design for a dialog to specify filter rules	205
5.39	Configured filter for the difference visualization	206
5.40	Four-layered interactive conflict management dashboard design	207
5.41	Conflict management dashboard: metamodel conflicts	208
5.42	Conflict management dashboard: model conflicts	209
5.43	Conflict management dashboard: model conflicts in context	209
5.44	Conflict management dashboard: conflicts resolution tasks	210
5.45	Conflict management dashboard: collaboration with distributed access rights	211
5.46	Screenshot of the user interface for a dialog to specify filter rules	212
5.47	Navigation controls and download menu of the MODELGLUE's visualizations	213
6.1	Architectural overview of MODELGLUE as UML component diagram	217
6.2	Access right dialog of a MODEL	219
6.3	Access right dialog of an OBJECTDEFINITION	219
6.4	Access right dialog of an ATTRIBUTEDEFINITION	220
6.5	Access right dialog of an OBJECT	221
6.6	Access right dialog of an ATTRIBUTE	221
6.7	A dialog to configure the differencing of MODELS	222
6.8	A dialog to configure a merge of MODELS	222
6.9	Tabular overview of model tasks within Tricia	223
6.10	Tabular view of conflict task details within Tricia	224
6.11	Architecture of the Visualization Component of MODELGLUE	226
6.12	Abstract visualization model for interactive visualizations	229
6.13	COLORGRADIENT with four COLORTRANSITIONS	230
6.14	Cutting literals within the visualization framework	232
6.15	Interaction model of the visualization framework	233
6.16	Interaction model of the visualization framework	236
6.17	InteractivePlanarSymbol within the interaction model of the visualization framework	237
6.18	Visitor pattern within the visualization framework	238
6.19	Overview of the Visualization Process as an Activity Diagram	241
6.20	Object graph for the approve interaction within the conflict management dashboard	243
6.21	Object graph for the validate interaction within the conflict management dashboard	244
6.22	Sequence diagram to generate user-specific visualizations with an Update according to [Sc13, p. 35]	245
6.23	Sequence diagram illustrating the broadcast of a configuration to generate user-specific visualizations to enable visual real-time collaboration developed with Schrade [Sc13, p. 38]	246

6.24	Sequence diagram illustrating the broadcast of a single visualization to enable visual real-time collaboration developed with Schrade [Sc13, p. 39]	247
7.1	Evaluation steps of MODELGLUE combining different research strategies	251
7.2	Core metamodels of the information sources [Ki14, p. 72]	256
7.3	Integrated current state of the EA [Ki14, p. 74]	257
7.4	Integrated planned state of the EA [Ki14, p. 75]	257
7.5	Visual representation of MODELELEMENTS with conflicts [Ki14, p. 81]	261
7.6	Adaptations of the differencing visualization in the three-way comparison [Ki14, p. 84]	264
7.7	Excerpt of the EA metamodel of the organization [Ki14, p. 95]	269
7.8	Different quantities of objects in the second and third layer of the visualizations [Ki14, pp. 78,98]	270
7.9	Background colors of the overlay windows within the visualizations [Ki14, pp. 93]	276
7.10	Exemplified MxL expressions [Ki14, pp. 66]	278
A.1	Core structure of the XML-Schema of a mapping in MODELGLUE	297
B.1	Mapping of MODELELEMENTS and ROLES to the prototypical implementation [Ki14, p. 21]	303
B.2	Mapping of OPERATIONS to the prototypical implementation [Ki14, p. 34]	304

List of Tables

1.1	Challenges organizations face during EA model maintenance [RHF ⁺ 13]	3
1.2	How organizations procure information for EA model maintenance [RHF ⁺ 13]	3
4.1	Use Case Template of Cockburn [Co01, ch. 11]	87
4.2	Use Case: Integrate Information Source	90
4.3	Use Case: Configure Mapping	92
4.4	Use Case: Synchronize Information Source	96
4.5	Use Case: Import Information	98
4.6	Semantic links vs. information import	100
4.7	Use Case: View Differences	101
4.8	Use Case: Merge Models	103
4.9	Use Case: Resolve Conflicts	106
4.10	Use Case: Resolve Conflicts Collaboratively	108
4.11	Use Case: Configure conflict resolution strategy	110
4.12	Overlapping changes according to Wieland et al. [WLS ⁺ 12]	111
4.13	Use Case: Assign Task	113
4.14	Use Case: View Conflicts	115
4.15	Mapping of use cases to requirements and respective categories	129
5.1	Mapping of the Ecore kernel and the core concepts of MODELGLUE	156
5.2	Outline of the compression method applied to changes	173
5.3	Merging cardinality constraints	192
7.1	Characteristics of different evaluation methods	250
8.1	Mapping of use cases to requirements and respective categories	291

Enterprises face highly competitive markets driven by enormous cost pressure. In response to globalization and industrialization, managers are forced to reengineer how their business operates [HC93] and how they can foster innovation [ABP06]. As a consequence, enterprises transform. Mutual alignment of business and Information Technology (IT) is a crucial factor for successful enterprise transformations [Ve94] to reduce time-to-market and to support new and existing capabilities by information systems in an effective and yet cost efficient manner. Inherently complex, ongoing trends such as hyperconnectivity [We01, QHCW05], cloud computing [MG11], and big data [BCM10] accelerate the design of new business models and open new possibilities but also aggravate the holistic management of an entire organization's IT. Although complexity increases, managers of today's leading enterprises are able to transform their business while creating innovative products and new business models outperforming their competitors [WR09, p.115ff]. To transform efficiently and to anticipate undesired side-effects of changes to business or IT, a holistic view of the organization is required.

A commonly accepted means to facilitate enterprise transformations is the strategic management of an Enterprise Architecture (EA) [RWR06, Ha10]. Schönherr [Sc08, Sc09] reports that there is no common definition of the term EA. Although there are plenty attempts to define the term, e.g. [Th11], in this thesis we consider an EA as a conceptual view of a software intensive system in line with the International Organization for Standardization (ISO) [In07] and our recent publications [BEG⁺12, FBH⁺13, RHF⁺13] (cf. Definition 1.1).

DEFINITION 1.1: Enterprise Architecture

An *Enterprise Architecture* is a comprehensive conceptual view on an organization as a whole. It embraces essential elements of the entire organization including relationships among them as well as to its environment. ■

Managing an entire EA is a challenging task [HSR⁺13]. The EA management discipline seeks to strategically plan and guide EA transformations and copes with resulting complexity. That includes enacting a managed evolution of an EA towards a desired visionary state. The evolution of an EA aims to align business and IT with each other and with the strategy of the organization [RWR06].

Primarily, EA management creates value by deriving planned states that outline EA transformations or by providing information to its stakeholders. The former seeks to operationalize the evolution of an EA by increasing either effectiveness or efficiency of an EA whereas the latter facilitates and ultimately accelerates or simplifies work of stakeholders. Both alternatives must be backed by a sound information base for profound decisions. In particular, planning transitions from the current state towards a desired target state of an EA requires an overview of the status-quo. This current state of an EA is commonly documented in the course of EA model maintenance endeavors [La13, p. 54].

Although it is possible to carry out this documentation in an informal manner, commonly EA information is captured formally, i.e. in the course of EA model maintenance endeavors, EA information is added to an *EA model*. An EA model describes a specific part of an EA in a formal manner. It contains EA information commonly conceptualized in a metamodel describing relevant entities and relationships among them. The purpose of an EA model is to serve as a solid basis for decision-making for EA management and its stakeholders. Thus, it is of high importance that the EA model reflects the reality in terms of desired granularity, correctness, topicality, and completeness [Ha10, p. 70].

Well-executed EA management is a strategic advantage [WR09, ch. 1]. However, often, large-scale EA management endeavors are financed without showing any substantial benefit to stakeholders of the EA management discipline. It is a considerable challenge for EA management endeavors to provide evidence for its benefits [Sc05, p. 59]; this holds true for young and unexperienced EA management endeavors in particular [HSR⁺13].

1.1 Problem Statement

Many authors report that the EA model maintenance process poses a challenge and is often regarded both, error-prone and time-consuming [BEG⁺12, FBH⁺13, HMR12, RHF⁺13]. Long-lasting EA model maintenance endeavors unable to show early results in a timely manner cannot provide the decision base required for adequate EA management. As a consequence, many EA management endeavors lack the ability to show an early Return on Investment (ROI) [AKL99b] or an ROI at all [OPW⁺09, p. 129]; a situation which often ends in stakeholder dissatisfaction [HSR⁺13] and a badmouthed EA initiative that is very likely to fail.

In collaboration with University of Innsbruck, we investigated the maintenance process of EA models empirically. In [RHF⁺13], we present results of a survey carried out among 140 EA practitioners; two key observations are central for the further discourse within the present thesis. First, Table 1.1 illustrates the challenges organizations face when maintaining an EA model. The figures confirm that maintaining an EA model is perceived as a huge effort while the outcome is dissatisfactory, i.e. data quality of the EA model often remains low. At

the same time, practitioners also report insufficient tool support. In [RHF⁺13], we present results of a statistical analysis of the two challenges ‘insufficient tool support’ and ‘huge data collection effort’. One of our hypotheses in [RHF⁺13] is that a successful EA model maintenance process requires adequate tool support. We applied Pearson’s chi square (χ^2) test [Pe00] to evaluate our hypotheses. 39 (~%81) of 48 participants reporting insufficient tool support also agree with the time consuming nature of EA model maintenance; 54 (~%58) of the remaining 92 participants which do *not* report inadequate tool support confirm the high efforts for EA model maintenance. Our null hypothesis is that as many participants confirm the ‘time consuming nature of data collection’ as the ‘inadequate tool support’. The result of the χ^2 goodness of fit test allows us to reject our null hypothesis with $p = 0.007$ ($p \leq 0.05$) [RHF⁺13]. Consequently, the analysis of our empirical findings strongly suggests that the effort of EA model maintenance depends on adequate tool support.

Challenge	n	% of all
Huge data collection effort	77	55.0%
Low EA model data quality	77	55.0%
Insufficient tool support	48	34.3%
No management support	44	33.4%
Low return on investment	36	25.7%
Other	32	22.9%
No specific challenge	10	7.1%

Table 1.1: Challenges organizations face during EA model maintenance [RHF⁺13]

Second, we investigated the sources utilized to acquire information relevant for EA management. Such an *information source* contains knowledge relevant for EA management or its stakeholders in a tangible or intangible manner. Table 1.2 shows different methods and techniques applied in organizations to procure information for EA management. We found that enterprise architects often utilize existing information systems, i.e. applications and databases, to gather information relevant for their EA model.

Type of collection	n	% of all
Manually from applications/databases	95	76.0%
Manually via interviews	85	68.0%
Manually modeled in workshops	66	52.8%
Manually via questionnaires	46	36.8%
Partially collected automatically	44	35.2%

Table 1.2: How organizations procure information for EA model maintenance [RHF⁺13]

Another hypothesis analyzed in [RHF⁺13] is that the use of automation techniques decreases required efforts for EA model maintenance. Our null hypothesis is that participants who have implemented automation techniques and those who have not equally complain about the time consuming nature of EA model maintenance. 64 (~70%) of 91 participants who have *not* implemented automation complain about the time needed to collect information.

1. Introduction

Only 12 (~44%) of 27 participants whose organization employs automation complain about this challenge. This can indicate that automation actually has a positive effect on the time spent for EA model maintenance. The goodness of fit test results with $p = 0.014$ ($p \leq 0.05$). Thus, our empirical results strongly suggest that automation techniques reduce efforts to maintain an EA model¹.

In line with these observations, recent research [BEG⁺12, FAB⁺11a, RHF⁺13] focuses on utilizing existing information sources within the organization to accelerate EA model maintenance. Although practitioners widely agree that EA management uses information that is often already contained in existing applications [FBH⁺13], these approaches face new challenges [HMR12].

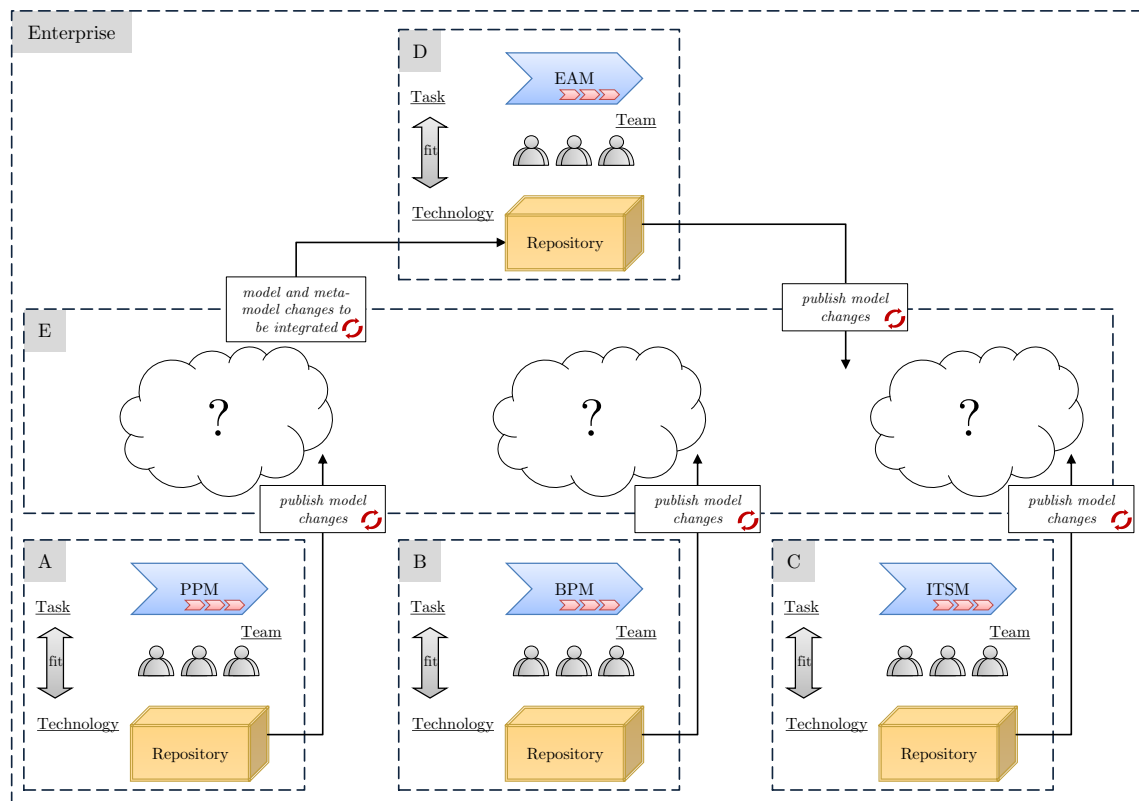


Figure 1.1: Information exchange in a federated EA model environment as a socio-technical system of systems

For our next considerations, Figure 1.1 illustrates an organizational structure and situation we often observe in practice. It shows specialized communities **A**, **B**, and **C**, their processes, involved teams, and repositories which are used by the teams to operate. In this context, a community can be regarded a business unit. Members of such a business unit perform tasks locally, i.e. tasks within the community. For the common good of the organization, community members share information by publishing their model changes (**E** in Figure 1.1). Goodhue and Thompson [GT95] observe that individual performance

¹We refer to [RHF⁺13] for a discussion on potential threats to validity and a comprehensive description of the dataset.

rises when a good fit between tasks and technology prevails and coin the term ‘*task-technology fit*’. We assume that the repositories of the specialized communities fit best for the tasks at hand in these communities.

ASSUMPTION 1.1: Modeling community members

We assume that although members of modeling communities are eager to accomplish priority objectives of their respective individual community primarily, they are willing to share information for the common good of their organization.

EA management (D in Figure 1.1) integrates this information in a coherent and holistic model maintained in a separate repository. Note that the three communities project portfolio management (PPM), Business Process Management (BPM), and IT Service Management (ITSM) serve as a vivid example. In Figure 1.1, we illustrate three rather concrete communities to describe a general structure between these communities and EA management; in practice we observe quite a few of these communities which—together with the EA management community—build a *federated EA model environment* (cf. Preliminary Definition 1.1).

PRELIMINARY DEFINITION 1.1: Federated EA modeling environment

A federated EA model environment refers to the evolutionary development and maintenance of a common model and metamodel that is based on

- autonomous,
- distributed, and
- heterogeneous

information systems. The common metamodel is called *federal metamodel* when existing information systems are utilized as information source in an automated fashion. A federal metamodel might be extended by additional concepts over time; its model is maintained (semi-)automatically and manually by end-users. □

Within such a federation, EA management exhibits the federal or central entity as it is the community that maintains an integrated decision base seeking to tie together pieces of different information sources in a coordinated manner. Thereto, EA management also establishes processes and performs tasks on a regular basis.

Symptomatic for such a federated EA model environment are model conflicts, which have to be resolved [RHM13b], e.g. if two communities describe the same real-world object. However, as of today, in EA management neither Buschle et al. [BHS⁺12, BEG⁺12], Farwick et al. [FAB⁺11a], nor Fischer et al. [FAW07] detailed how to resolve these model conflicts. We further diagnose that it remains unclear how information of specialized communities is exchanged with EA management as well as if and how this process can be facilitated by a socio-technical solution. In [HMR12], we report on these challenges based

on our practical experience and results of a survey among 123 experts in the field of EA management and IT management. In the paper, we provide four categories of challenges: (model) transformation challenges, data challenges, business and organizational challenges, and tooling related challenges. These categories give a first impression on the socio-technical context of a federated EA model environment and confirm a significant research gap.

In the present thesis, we investigate on the characteristics of a federated EA model environment, coin the term *Federated EA Model Management*, and provide software-support for Federated EA Model Management. In the next section, we present research questions to guide this investigation.

1.2 Research Questions

After outlining the problem investigated in the present thesis, we deduce research questions that guide the investigations on the management and maintenance of an EA model in a federated EA model environment. With our first research question, we intend to describe the current situation in organizations so far referred to as federated EA model environments.

Research question 1 (RQ1): What are typical characteristics, i.e. core use cases, involved roles, and information sources, of a federated EA model environment?

Our recent empirical observations [RHF⁺13] reveal that EA management commonly integrates different knowledge-bases in a coherent model to serve as a single point of truth for decision makers. For this purpose, EA model maintenance endeavors utilize both, EA information gathered manually from stakeholders or automatically from information sources within operative IT environments. Further, we observe that stakeholder demands may vary over time [Ha13] such that an adaptation of the EA metamodel is needed and other, not yet considered information sources could become relevant [RHM13a]. In the course of this thesis, we underpin that EA model maintenance is not a one-off endeavor but an iterative process [BMM⁺11a]. It takes place continuously to provide up-to-date information to decision makers in a timely manner. This includes an extension of the EA metamodel over time, which is required to document new concepts, attributes, and relationships. Utilizing the knowledge of the characteristics of a federated EA model environment, our intent is to improve the current situation. Hence, we raise the next research question:

Research question 2 (RQ2): How can a system design and a process design that continuously integrate models from specialized and autonomous modeling communities with an EA model look like?

Such an integration includes the detection, communication, and resolution of conflicts between different models. Especially the resolution takes place across different communities. Since these communities want to be in full possession of their repository, a system facilitating Federated EA Model Management must be capable to manage multiple metamodels of different information sources. Hence, the system must feature a metamodel that is able to

homogenize metamodels. Models as well as metamodels of information sources may change over time. Concurrent modeling is an inherent characteristic of the continuous integration process of information such that in a federated EA model environment, model conflicts such as concurrent updates or deletes are inevitable. Thus, the metamodel must allow collaborative conflict management among multiple modeling communities. The development of a metamodel suitable for Federated EA Model Management is addressed by our next research question:

Research question 3 (RQ3): How can a metamodel look like that realizes Federated EA Model Management?

Striving for consistency, community members seek to resolve conflicts, such as contradicting information, to reach a common agreement. On the other hand, involved parties could also agree to maintain local inconsistencies between the different repositories. Communicating, managing, and resolving model conflicts can be facilitated by an information system. Put in context, model conflicts can be regarded as complex information. Few [Fe06, ch. 1] argues that a dashboard can be used to communicate a vast amount of information. He further outlines that details are often shown through splitting up the dashboard and enabling interaction [Fe06, sec. 3.1.1]. We conclude that finding a suitable user interface (UI) is an important factor for a high user adoption.

Research question 4 (RQ4): What is a suitable integrated UI for collaborative Federated EA Model Management?

Especially when realizing a system with sophisticated UI support, further challenges arise. Additionally, aspects in the course of concurrent modeling in a federated EA model environment are subject of our next research question. With this question, we seek for an answer how our conceptual work can be realized by an implementation artifact:

Research question 5 (RQ5): What are the software engineering challenges for a system that facilitates Federated EA Model Management?

According to Hevner et al. [HMP⁺04], our concepts must be evaluated and iteratively refined based on practitioner feedback to prove the utility of the solution in a given context. In order to evaluate our approach, its practical application is subject of the final research question:

Research question 6 (RQ6): What is the experience of real stakeholders using this system of systems for EA model maintenance? What are the specificities and further challenges of the Federated EA Model Management process?

1.3 Research Design

In this section, we outline the research approach taken. The research design of the thesis pursues the conceptual framework for design science research² in information systems presented by Hevner et al. in [HMP⁺04].

Design science is concerned with the creation and evaluation of new artifacts. While the evaluation ideally should be a result of practical application in an organizational setting, the creation of an artifact should “extend the boundaries of human problem solving and organizational capabilities by providing intellectual as well as computational tools” [HMP⁺04]. That is, theories are not only developed by observation and synthesis of best-practices but are gained through experiments in a practical setting. This includes constructs, models, methods, or instantiations [MS95]. An assessment of the created artifact is based on the utility it provides to solve identified organizational problems.

The conceptual framework presented by Hevner et al. intends to provide guidance for all information systems research for understanding, executing, and evaluating the research. Figure 1.2 illustrates the framework.

Fundamental property of the framework is its iterative nature, i.e. a repeated loop between two phases: the creation or refinement and the assessment of an artifact. Hevner et al. refer to these phases as ‘*develop/build*’ and ‘*justify/evaluate*’. Hevner [He07] summarizes these phases of the information systems research as the ‘*design cycle*’.

As depicted, information systems research interacts with the environment and the knowledge base. Within the environment are people, organizations, and technology. On the one hand, it represents the problem space (cf. [Si96]) and ‘*business needs*’ addressed by the conducted research. On the other hand, the environment defines the context in which an artifact is applied and assessed. Practical requirements for an artifact originate from the environment. The assessment of the artifact in the environment builds the context of the evaluation. This relationship is referred to as ‘*relevance cycle*’ in [He07].

In the course of both phases of the design cycle the ‘*raw materials*’ of the existing information systems knowledge base are applied. That means results of prior information systems research, e.g. methods and models, are applied to create or refine a new artifact in the ‘*develop/build*’ phase. Methodologies serve an appropriate assessment of the utility of the artifact and provide guidelines to evaluate and justify the artifact. Rigor is achieved by selecting applicable knowledge appropriately to build and evaluate the artifact. Outcomes of the conducted research contribute to the knowledge base and may inform the community about future research. In [He07], Hevner summarizes the selection and contribution to the knowledge base as the ‘*rigor cycle*’.

Hevner et al. establish seven guidelines “to assist researchers [...] to understand the requirements of effective design-science research” [HMP⁺04]. Below, we outline these guidelines briefly and describe how they were taken into account in the research presented in this thesis.

²For an overview of different research methods in information systems, we refer the interested reader to [Fr06, ch. 7]. In his technical report, Frank also introduces a framework to configure different research methods [Fr06, ch. 9].

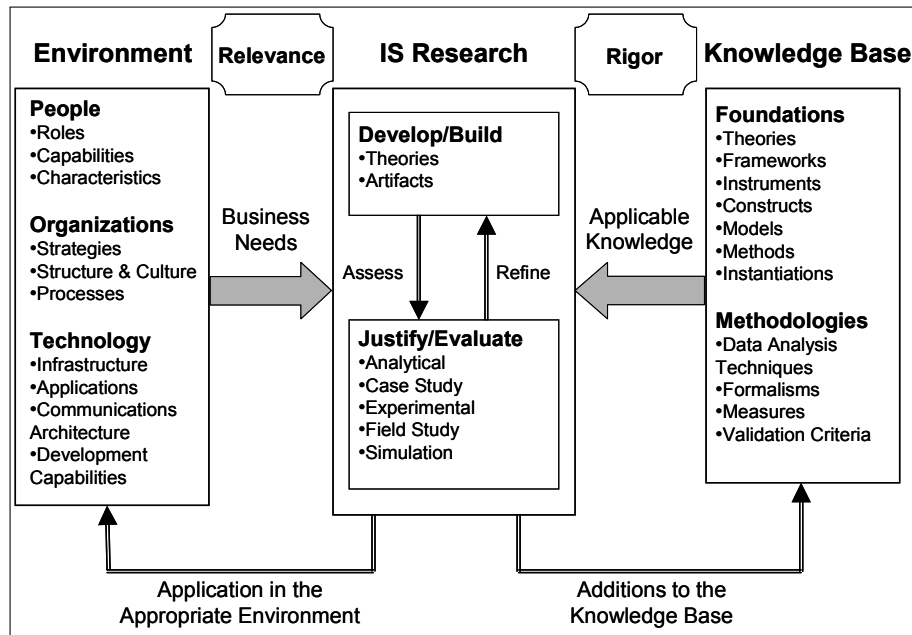


Figure 1.2: Information systems research framework of Hevner et al. [HMP⁺04]

1. **Design as an artifact.** The first guideline is summarized by Hevner et al. as follows. “Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation” [HMP⁺04]. In this thesis, an approach for a federated management of EA models is developed. Besides its conceptual description, also a viable software prototype was implemented. Hevner et al. calls this an *instantiation* of an artifact.
2. **Problem relevance.** The second guideline of Hevner et al. states information system research has to develop technology-based solutions that address important and relevant business problems. The goal of this thesis is to develop an approach to facilitate Federated EA Model Management to improve the current situation in EA model maintenance endeavors of organizations. Related problems to EA model maintenance endeavors as well as to federated approaches to information management are well studied (see Section 1.1, Section 2, and Chapter 3). In 2011, we gained first project experiences facing problems related to this thesis. First results reporting on initial findings were published in the beginning of 2012 [BEG⁺12]. To investigate problems in EA management on a broad scale organizational factors that influence EA management endeavors were investigated in [HSR⁺13]. In particular, we looked at factors and challenges organizations face that are related to automated approaches for EA model maintenance [HMR12]. The data quality aspects of one particular information source were investigated in [GMR12]. A variety of typical information sources were analyzed in [FBH⁺13].
3. **Design evaluation.** Hevner et al. argue that the “utility, quality, and efficacy of a design artifact must be rigorously demonstrated” [HMP⁺04]. The authors suggest a variety of design evaluation methods split into five categories, i.e. observational,

analytical, experimental, testing, descriptive. Within the observational methods, the case study is described as the study of an “artifact in depth in [a] business environment” [HMP⁺04]. For the evaluation of MODELGLUE, we apply observational methods in Chapter 7. We carried out two case studies during which we tested the prototypical implementation with industrial data and gathered feedback from practitioners. Additionally, we carried out an interview series in order to get feedback for our concepts from a broader audience and to foster an understanding of a viable Federated EA Model Management process.

- 4. Research contributions.** The core contribution of this thesis is the design artifact, i.e. MODELGLUE, a software-supported process design for Federated EA Model Management that improves the current situation for EA model maintenance and allows to increase the model consistency in an enterprise collaboratively. The core use cases of Federated EA Model Management are described in Chapter 4 whereas the design of MODELGLUE is detailed in Chapter 5. Technical feasibility of the approach is demonstrated by implementing a prototype based on an existing enterprise 2.0 collaboration platform (see Chapter 6). Utility of the prototype is shown in an empirical assessment in the context of the case studies whereas utility of the overall process that guides MODELGLUE is confirmed in an interview series (see Chapter 7).
- 5. Research rigor.** Hevner et al. state that “design-science research relies upon the application of rigorous methods in both the construction and evaluation of the designed artifact” [HMP⁺04]. The foundations of Federated EA Model Management and related fields such as EA documentation, software and model merging, and federated information systems were extensively studied. We report on the state-of-the-art in EA model management in Chapter 3. Further, the design goals and the guidelines followed during the design of MODELGLUE were precisely formulated (cf. Chapter 4 and Chapter 5). Finally, the arrangements, execution, and results of the evaluation are described comprehensively in Chapter 7. That includes details of in-depth interviews and a detailed description of the case studies.
- 6. Design as a search process.** Hevner et al. consider the design of an artifact as a search process. “The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment” [HMP⁺04]. MODELGLUE is the result of an iterative process that can be roughly divided into three phases. In the first phase, we obtained an understanding of current practices and point out problems in the domain of EA model maintenance. In the second phase, we attempted to establish concepts for an automation of EA model maintenance and we gathered feedback from industry for single components of MODELGLUE, e.g. processes, conflict types, conflict visualizations and its technical basis. A third and final phase was conducted when it turned out that these components seemed to be promising solution artifacts, the components were integrated in a coherent solution design as a single artifact named MODELGLUE.
- 7. Communication of research.** The final guideline of Hevner et al. proposes to present research to both, “technology-oriented as well as management oriented audiences” [HMP⁺04]. The former is addressed by providing implementation details and demonstrating that MODELGLUE can be integrated in an existing enterprise collaboration platform. The conceptual design as well as industry feedback obtained during the

case studies and interview series serve the latter. Preliminary results have been published to more business-oriented venues, e.g. [BEG⁺12, FBH⁺13, RHF⁺13, RHM13b], as well as technical venues, e.g. [HMR12, HRP⁺13b, RM14, RHM13a], and are references in the remainder of this thesis.

1.4 Contributions

The research questions and the research design aim to guide our research which contributes to the knowledge base regarding collaborative management of EA models in federated modeling environments. Towards this objective, we propose MODELGLUE. MODELGLUE fosters the continuous efforts to increase integrity and consistency in federated EA model environments. Subsequently, we outline the main contributions of the present thesis and MODELGLUE (cf. also Figure 1.3).

The first contribution is a **comprehensive description of the characteristics and challenges of Federated EA Model Management** based on available literature in related fields and empirical studies within the domain of EA management. We employ a **topic map** (A1) for EA model maintenance to revisit the state-of-the-art in literature and point out **research gaps** (A2). Essentially, a federated EA model environment is an organizational setting in which a central architecture management function integrates models of specialized, semi-autonomous IT management functions. We describe **typical characteristics** (A3) such as relationships between involved models, stakeholders, and roles. Additionally, we give an overview of data quality aspects of relevant information sources. Based on these characteristics, we analyze the **core use cases** (A4) which build the foundation for **requirements** (A5) of a software-supported process design.

Our second contribution is a **holistic design and implementation of a system of systems for Federated EA Model Management**, which build the foundation of a consistent modeling environment to develop a central, integrated model among multiple distributed, autonomous, and heterogeneous information systems while maintaining federated models of these information systems in a collaborative manner. We present a **system design** and a **process design** (A6). A *metamodel* (A7) realizing a dynamically-typed repository builds the groundwork of our design. We extend entity types, relationship types and attributes types of an existing metamodel. We equip these types with roles such that the resulting metamodel features a fine-grained access control. The metamodel further incorporates tasks and transient changesets to facilitate collaboration among different roles when model conflicts arise. These tasks are used to realize a **flexible, collaborative, and conflict-driven model evolution process** that guides the resolution of model conflicts. In this process, users of MODELGLUE receive tasks to manipulate the life-cycle of model elements. The life-cycle of a model element is captured in its state. MODELGLUE utilizes the fine-grained access model to realize a role model and to provide escalation mechanisms. This way, different stakeholders can be involved in the conflict resolution process. We present an implementation of MODELGLUE in a state-of-the-art Enterprise 2.0 framework, **mechanisms and algorithms** (A8) for model and metamodel (local and cross-model) conflict detection as well as **strategies for semi-automatic model conflict resolution** (A9) through conflict resolution tasks. These are integrated in a conflict management dash-

1. Introduction

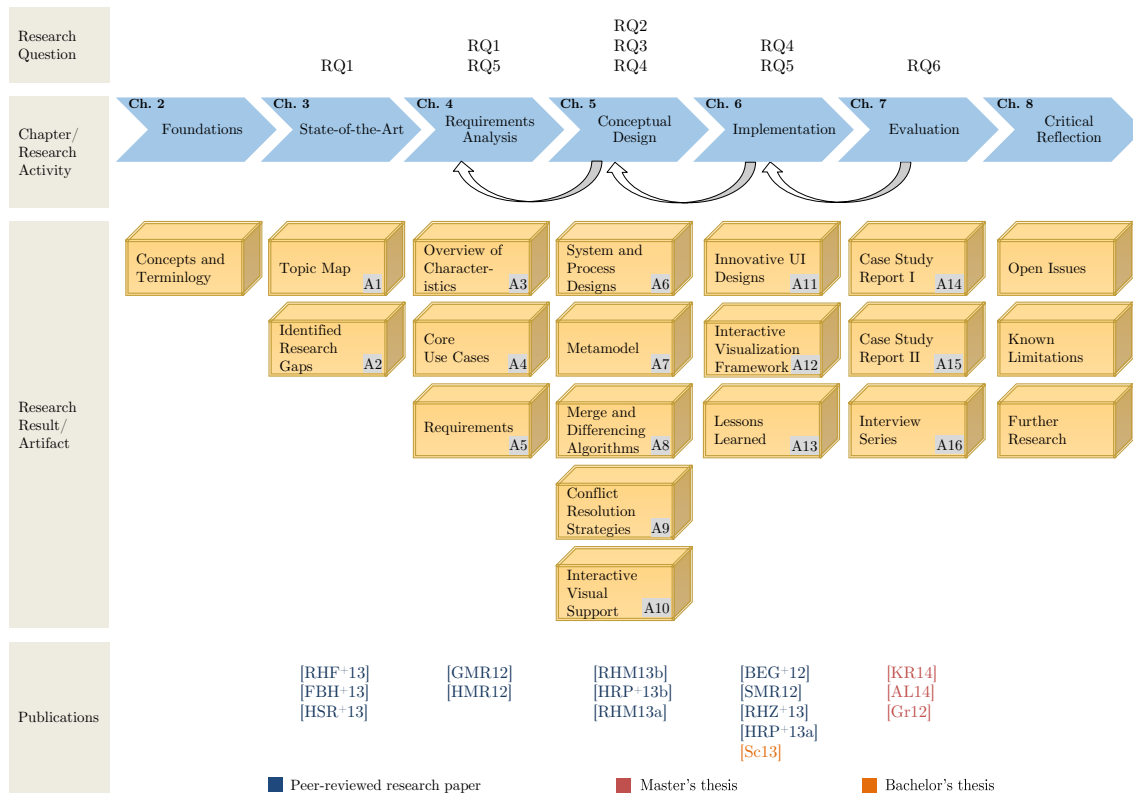


Figure 1.3: Research questions, structure of the thesis, and related publications

board that allows resolving conflicts collaboratively and is part of the **interactive visual support** (A10) of MODELGLUE. The dashboard is used to communicate conflicts visually and it allows the different roles involved to resolve model conflicts interactively. Moreover, MODELGLUE facilitates model quality assurance by providing visual means to view model differences. Besides **innovative UI designs** (A11), we provide implementation details of an **interactive visualization framework** (A12) and present **lessons learned** (A13) during the iterative development of a prototypical implementation of MODELGLUE.

Our third contribution is a **practical evaluation** of MODELGLUE in industry to provide feedback on the developed solution. We present an overview of published artifacts that prove feasibility and continue to summarize learnings from **two case studies** (A14 and A15) that use industrial data. The case studies embrace models of the application landscape and the infrastructure configuration management database (CMDB) of two German organizations using live data captured over a time period of 52 months and 1 month respectively. We utilize multiple physically federated repositories and integrate these with a single logically federal EA repository. During the case studies, we extended and refined our design iteratively. The present thesis reports on the final design and details rationales for the design decisions as well as practitioner feedback. A subsequent **interview series** (A16) among 11 EA experts confirms the general design and flow of events, i.e. applied methods and processes when pursuing Federated EA Model Management. The results of both, the case studies and interview series, give further insights how to pursue Federated EA Model Management.

1.5 Structure of the Thesis

This thesis is divided into eight chapters. Figure 1.3 illustrates the structure of the thesis, the addressed research questions, created artifacts as outlined above as well as related core publications. Each chapter is outlined in the following.

Chapter 1—Introduction—motivated the present thesis, detailed the problem, and derived research questions which guide the present thesis. Further, we described the research design taken and outlined the core contributions. The chapter concludes with this outline.

Chapter 2—Foundations—summarizes existing work relevant for the remainder of the thesis. The chapter serves the reader to build foundations to understand the thesis and refers to literature relevant for the topics investigated.

Chapter 3—State-of-the-Art in EA Model Maintenance—presents a topic map outlining gaps identified throughout our research and revisits the state-of-the-art referring to work of others that influenced our design decisions.

Chapter 4—Requirements Analysis—characterizes a federated EA model environment, details core use cases relevant for Federated EA Model Management, and derives requirements for an information system that supports Federated EA Model Management.

Chapter 5—Federated EA Model Management Design—presents the conceptual design of Federated EA Model Management and MODELGLUE. That embraces methods, techniques and a design for the software-support of Federated EA Model Management.

Chapter 6—Software Support for Federated EA Model Management—reveals implementation details of MODELGLUE. A particular focus is put on collaboration aspects and visual support. As a consequence, we outline a framework for interactive visualizations and demonstrate how we apply it to realize our conceptual design of MODELGLUE in a prototypical implementation.

Chapter 7—Evaluation—reports on the setup and results of two case studies in an industrial setting. In the chapter, we further present qualitative insights from a broader audience and feedback as an outcome of an interview series.

Chapter 8—Conclusion—summarizes the thesis' contributions, critically reflects on the contributions and the results, informs about further research, and finally outlines parts of the thesis that potentially could be generalized by subsequent investigations.

In this chapter, we clarify the important concepts relevant for the present thesis. Figure 2.1 depicts the diversity of related disciplines for this thesis as well as their intersections as a Venn diagram. In the following we briefly provide rationales for these different areas and outline their interrelationship, common ground, and relevance to this thesis.

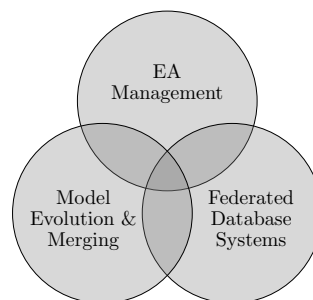


Figure 2.1: Intersection of different areas of related work for the present thesis

Areas summarized cover topics of different information systems and computer science communities, namely:

- **EA management** is the *problem domain* in which we observe insufficiencies concerning methodological guidance and tool support in the initial phase and subsequent maintenance of EA models. To provide a general understanding of the domain, we describe fundamental concepts and principles of EA management upon which we build our considerations in the remainder of the present thesis.

- **Model Evolution & Merging** also concentrates on the maintenance of models in a slightly different context. While EA models are formally defined, they are primarily used for knowledge management and decision making. In contrast, the model evolution and merging community is mostly interested in merging executable models commonly used in model-driven engineering (MDE). “Global model management involves keeping track of the relationships between various models, and, from time to time, combining information from several models into a single model” [BCE⁺06].
- **Federated Database Systems** focus on storing and sharing information in a federated structure between different databases. This field is also concerned with the exchange of information while federated systems, i.e. databases, stay autonomous.

Next, we sketch the problem space, i.e. fundamental concepts that make up an EA and the different perspectives on EA management and modeling.

2.1 EA and EA Management

In this section we introduce concepts which are central for EA management endeavors and illustrate the relationships between these concepts. After introducing our perspective of an EA as a whole, we revisit the core contribution of the ISO 42010 standard briefly. The conceptual model of the ISO serves to explain important relationships among concepts. We then put focus on the interdisciplinary role of EA management and its relationship to other management disciplines. This aspect of EA management plays an important role in the discourse of this thesis. Finally, a high-level overview of the phases of EA management is shown to point out an important aspect, i.e. the stakeholder involvement and iterative and incremental nature of EA management in general and in particular the impact on the maintenance and evolution of an EA model.

2.1.1 A Structural Perspective on an Enterprise Architecture

We observe that many EA researchers [RWR06, p. 47] and practitioners [Ha10] have a similar view on an enterprise¹. In line with Buschle et al. [BEG⁺12], Doucet et al. [DGS⁺09, pp. 79,345], Hanschke [Ha10, p. 66], Matthes et al. [MBL⁺08, pp. 28–31], Winter and Fischer [WF07], and Wittenburg [Wi07] we illustrate a high-level, holistic view on an enterprise in Figure 2.2.

It is divided in different layers and cross-functional aspects which exert influence on all layers. In the following we explain the layers from top down.

- **Business Capabilities** “describe the functional building blocks of an enterprise’s business model as part of the business architecture” [Fr14, p. 2]. They are core abilities of an enterprise (cf. Prahalad and Hamel [PH90]).

¹Note that other research communities describe an organization from a different perspective; for instance Inkpen and Choudhury [IC95], Morgan [Mo83], and Weick [We12, p. 3ff].

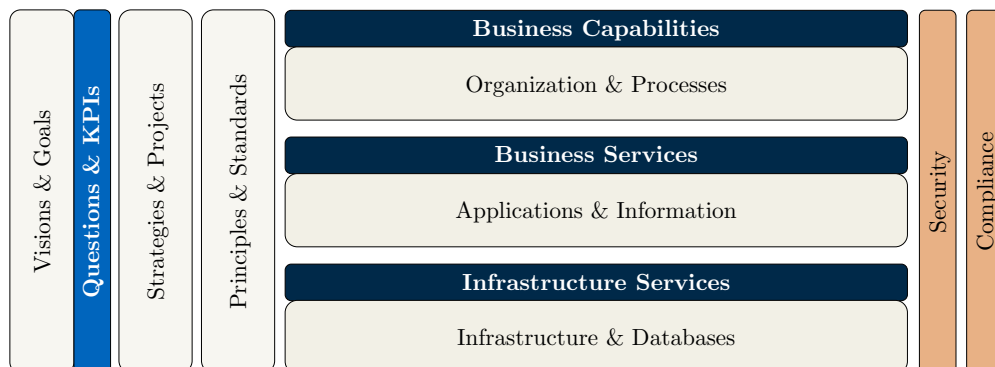


Figure 2.2: Fundamental structure of a holistic view on an enterprise [BEG⁺12]

- **Organization & Processes** are used to realize and implement business capabilities. The organizational structure, e.g. a matrix organization [Ga71], tends to be static while processes describe behavioral aspects for value creation (cf. Davenport and Short [DS90]). At this level, coarse-grained information is modeled such as business objects, e.g. customer, product, or contract.
- **Business Services** describe the provision of tangible products or services to accomplish certain processes. The right granularity of these services enable reuse and an effective composition to new processes as a response to a frequently changing environment.
- **Applications & Information** describe the IT support and required information to accomplish a business service. This embraces not only in-house IT but also applications hosted by a third party, e.g. through Software as a Service (SaaS).
- **Infrastructure Services** are technical services used to provide applications and information, e.g. access to the Internet, the provision of an application server, etc. Similar to the applications, procurement of third-party services has become a commodity through Infrastructure as a Service (IaaS) and Platform as a Service (PaaS).
- **Infrastructure & Databases** is the technical backbone of an organization and includes for instance network elements like hubs, routers, and switches as well as servers and clusters. Databases on the other hand are the physical data storage for the most valuable asset today's knowledge-intensive organizations possess—*information*.

Other descriptions of these layers exist. To name one prominent example, ArchiMate 2.0 [Th12a, p.6] refers to these layers as '*Business Layer*', '*Application Layer*', and '*Technology Layer*'. As stated above, these layers are influenced by cross-functional aspects which are driving forces to any of these layers. These aspects are:

- **Visions & Goals** are derived from the enterprise strategy. Visions are desired states of the reality in the far future. Visions are operationalized to goals.

- **Questions & KPIs** help to measure the accomplishment of goals in a quantitative manner. Metrics help to formalize questions and key performance indicator (KPI) values can provide a means to control efforts.
- **Strategies & Projects** are instruments to implement change and to create innovation. Especially, projects transform the EA whereas strategies provide courses of action to achieve goals [Ob13, p. 29], [BMR⁺10a].
- **Principles & Standards** define guidelines and borders for the EA. Thus, they can be viewed as constraints for the solution space [BMR⁺10a].
- **Security** aspects are relevant to prevent e.g. industrial espionage, data loss, and contempt of private and personal data.
- **Compliance** relates to influencing factors like regulatory changes or audit trails (cf. [AHR14]).

In line with these observations, Winter et al. [WF06] consider an EA as depicted in Figure 2.3. To a large extent, this perspective is in line with the situation described in Section 1.1.

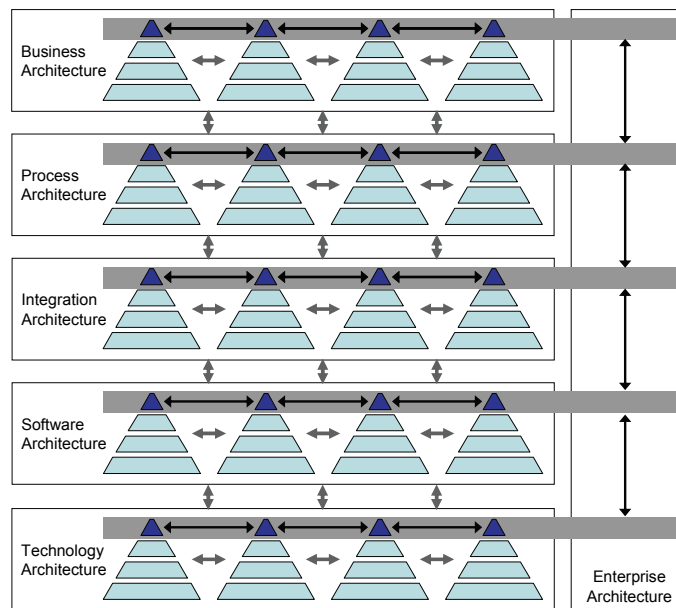


Figure 2.3: An EA as cross-layer view of aggregated artifacts [WF06]

We summarize key aspects of this perspective as follows:

- An EA and respective model is a description of an integrated view upon (aggregated) information of different architectural layers.
- The EA management function is primarily interested in the relationships between the different layers.

- A modeling community owns different information silos; these may be integrated.
- Any specialized knowledge and information remains within the respective architecture (cf. modeling community Section 1.1).

As outlined above, at a high level EA researchers as well as practitioners have the same or a very similar view on an enterprise. However, in detail organizations vary strongly when it comes to specifying the exact terminology, concepts, and their interrelationships, cf. [SW09]. This has a great influence on the metamodel of an EA model.

PRELIMINARY DEFINITION 2.1: Metamodel

A metamodel is a conceptualization of a class of models. It can be considered a language that determines how to specify permissible models [Fr11, p. 39]. □

To define a metamodel, commonly a (semi-)formal notation is employed. Prominent examples for these *languages* are:

- the Unified Modeling Language (UML) defined by the Object Management Group (OMG) [OMG11b],
- the Entity Relationship Model (ERM) introduced by Chen [Ch76], or
- the Object-Role Modeling (ORM) developed by Halpin [Ha05] (see also [HM10]).

Developing an EA metamodel that finds acceptance within an enterprise is a task influenced by many factors. Some important factors are intangible, others are described by literature. In the following, we outline a particular framework that the most prominent EA frameworks, researchers as well as practitioners often refer to.

2.1.2 Conceptual Framework of the ISO 42010 Standard

An EA can be considered a model of a software-intensive system (cf. [Th11, p. 9], [La13, p. 23]). These software-intensive systems can be described utilizing the conceptual framework as proposed in the ISO 42010 standard² [ISO07b]. Figure 2.4 depicts the framework for architectural descriptions of software-intensive systems, concepts therein, and their relationships to each other.

A MISSION is (directly or indirectly) derived from the business strategy³ and contributes to an objective, i.e. it should describe “a use or operation [...] to meet some set of objectives” [ISO07b]. One or multiple MISSIONS get fulfilled or supported by a software-intensive SYSTEM. We regard such a SYSTEM as a *system of systems*, i.e. an enterprise. An enterprise

²The ISO 42010 standard adopted the Institute of Electrical and Electronics Engineers (IEEE) standard 1471 [IE00].

³In [Po80], Porter presents three general widely accepted strategies. These are cost leadership, differentiation, and focus. As business strategies are beyond the scope of the present thesis, we refer the interested reader to [TW93, TW97] for other business strategies.

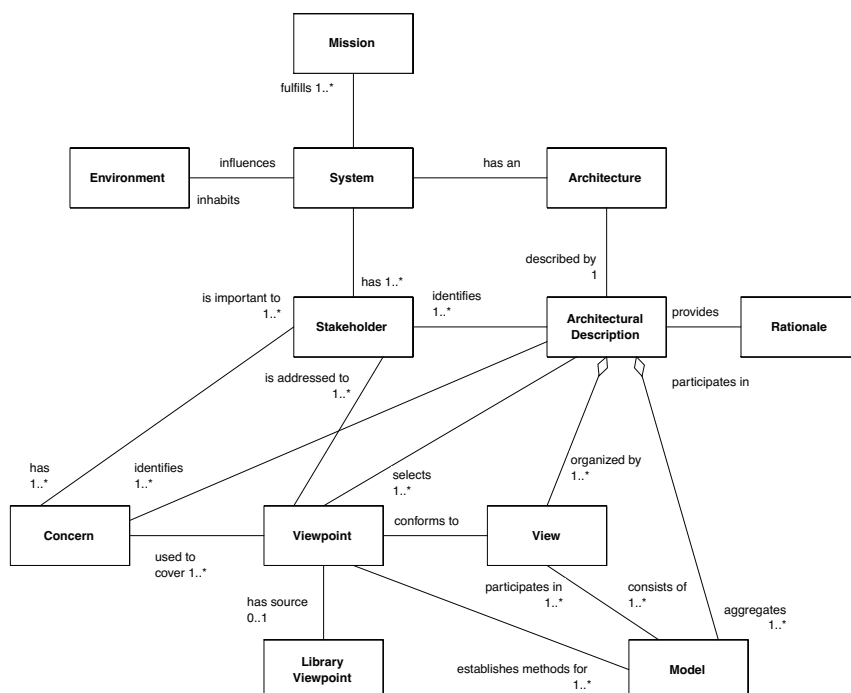


Figure 2.4: Conceptual framework of the ISO 42010 standard for architectural descriptions of software-intensive systems [ISO07b]

or **SYSTEM** is surrounded by an **ENVIRONMENT**. Consequently, a **SYSTEM** has accompanied properties of its **ENVIRONMENT**; the **SYSTEM** is influenced by the **ENVIRONMENT** and in turn influences it (cf. [Be68]).

An enterprise, as a system of systems, has individuals or groups of individuals that have interests or concerns regarding the system; these are called **STAKEHOLDERS**⁴. **CONCERNS** describe any aspects important to one or more **STAKEHOLDERS**, e.g. “considerations such as performance, reliability, security, distribution, and evolvability” [ISO07b].

As we consider the **SYSTEM** to be an enterprise and each system has an **ARCHITECTURE**, the respective architecture is an EA (cf. Definition 1.1). The EA can be documented by an **ARCHITECTURAL DESCRIPTION**. Note that the **ARCHITECTURE** is purely conceptual whereas the **ARCHITECTURAL DESCRIPTIONS** are concrete artifacts. The **ARCHITECTURAL DESCRIPTION** provides **RATIONALE** for the selection of a architectural concepts. This (enterprise) **ARCHITECTURAL DESCRIPTIONS** are maintained and analyzed using different **VIEWS**. A **VIEW** addresses particular **CONCERNS** expressing the **ARCHITECTURE** from a particular angle, i.e. a **VIEWPOINT**. This **VIEWPOINT** is “a specification of the conventions for constructing and using a view. A pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis” [ISO07b].

⁴The term stakeholder and its roots in management literature is investigated in depth by Freeman [Fr10, p. 32ff].

Both, the VIEW and VIEWPOINT can be described with a MODEL. Thus, MODELS (cf. Section 2.2.1) are means to describe the ARCHITECTURAL DESCRIPTION, VIEWS, and VIEWPOINTS; a LIBRARY VIEWPOINT is a VIEWPOINT that does not originate from the ARCHITECTURAL DESCRIPTION.

Above concepts and in particular their interrelationships serve to build a general understanding of the interactions within an EA viewing it as a software-intensive system. During the discussions in the remainder of this thesis, we refer to the relationships between STAKEHOLDERS their CONCERNS, VIEWPOINTS and VIEWS on an ARCHITECTURAL DESCRIPTION.

The ISO 42010 standard gives guidelines which describe the interrelationships in a conceptual model. In the following, we employ a different perspective to describe the relationships between EA management and other management disciplines.

2.1.3 EA Management Interrelated with other Management Disciplines

In the conceptual model of the ISO 42010 standard (cf. [ISO07b] or Section 2.1.2) the enterprise and the EA respectively are part of an environment. We already motivated that an EA is administered by an EA management function (cf. Chapter 2.1). Also, we outlined that an EA embraces business as well as IT aspects (cf. Section 2.1.1). EA management serves as a mediator between different management disciplines. With respect to a system approach, the environment of EA management can be considered other management disciplines. Wittenburg [Wi07, p.12] and Matthes et al. [MBL⁺08, p.28] highlight the importance of an integration with other disciplines for successful EA management initiatives. While we look at EA management as a continues process (cf. Section 2.1.4), we also see an integration with other management disciplines. This integration can be described taking a more static perspective. Figure 2.5 illustrates this integration with other disciplines.

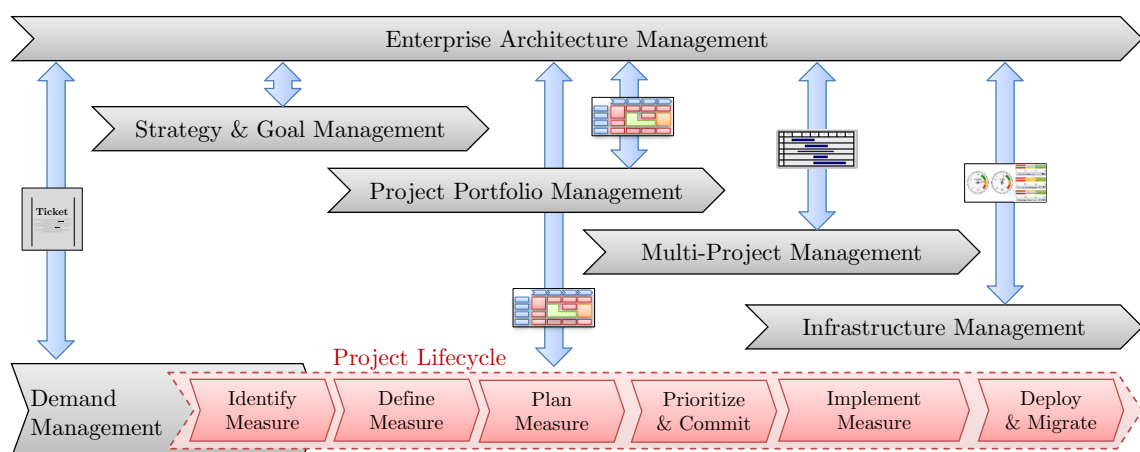


Figure 2.5: Integration of EA management with other management functions (based on [Wi07, p.12])

We stress that EA management must be linked to related management processes to use their input and output to plan, control, and monitor the evolution of the EA.

Figure 2.5 shows the different management areas and illustrates their interconnections referring to information flow and collaboration. In the following we sketch these management areas briefly:

EA management functions as a mediator between the different management areas by providing current, planned, and target states of the EA to align, plan, and control its evolution.

Demand Management is the entry point for new action items, which may change the EA in terms of projects. Typical tasks of demand management embrace gathering demands, identifying affected architectural elements, and preparing the transformation of project proposals.

Strategy & Goal Management puts focus on the realization, monitoring and evaluation of defined strategies and goals. Strategy and goal management assesses demands, projects, and their affected architectural elements with respect to the strategies and goals defined for the enterprise.

Project Portfolio Management uses project proposals to compose a project portfolio for the next planning period according to a set of criteria, e.g. [Kr10, pp. 258–261]:

- risk (project duration, project size, available resources, problem dimension, dependencies),
- utility (economic viability, lifetime, non-quantifiable utility, employee-orientation, potential development),
- strategic fit, (customer-orientation, competitor-orientation, process-orientation, efficiency), and
- application landscape plan, (process organization, process responsibility, process goals, information system architecture data, information system architecture functionalities, IT strategy and technology fit).

Multi-Project Management seeks to synchronize projects. According to project interdependencies derived from the objects used, e.g. application systems or services, a project is likely to change. Further, delayed projects may affect other projects. These deviations have to be identified and projects have to be rescheduled accordingly.

Infrastructure Management focuses on the provision of essential components required for IT operations such as e.g. hardware and other (typically physical) devices (cf. [Kr10, p. 272]).

Project Lifecycle describes the project execution phases that realizes change in an EA. Note that Figure 2.5 is a schematic view illustrating a sequential design process, e.g. the waterfall model [Ro70], and iterative and incremental approaches, e.g. agile projects, shall be referred to as well.

Typical information exchanged between the disciplines embraces e.g. the current, planned, or target state of the EA, project proposals as well as strategies and goals. Commonly, EA management serves as a mediator between these different management disciplines when exchanging and sharing information about an EA. Federated EA Model Management accounts for this fact. In this thesis, we consider these management functions as modeling communities and describe a design aiming at the utilization of existing information sources employed by these communities.

As of now, we viewed at an EA from a structural and more static perspective. Additionally, we outlined the interrelationships between the respective management function, EA management, and other management functions. In the following, we take a process perspective and describe high-level phases EA management endeavors typically carry out.

2.1.4 EA Management is a Continuous Process

Academia, e.g. Ahlemann et al. [ASM⁺12, p.249] and Buckl [Bu11, p.152], as well as practitioners, e.g. Hanschke [Ha10], Keller [Ke12], and Niemann [Ni05, p.38], describe EA management as an iterative process. In [BMM⁺11a] we first discussed an agile design which incorporates the basic ideas of iterations that deliver increments of an EA artifact. On this basis, we present a holistic design for an EA management process in [RZM14, p.12] and provide empirical results on the agility of current EA management practices [HRS⁺14].

Figure 2.6 depicts the EA management discipline as an iterative, incremental, and continuous process. The central notion of continues improvement through learning from past decisions (cf. Deming[De82]) often found in contemporary process-based frameworks, e.g. Information Technology Infrastructure Library (ITIL) [Ca11a, Ca11e, Ca11c, Ca11f, Ca11d] or The Open Group Architecture Framework (TOGAF) [Th11, p.445], has been incorporated in this process.

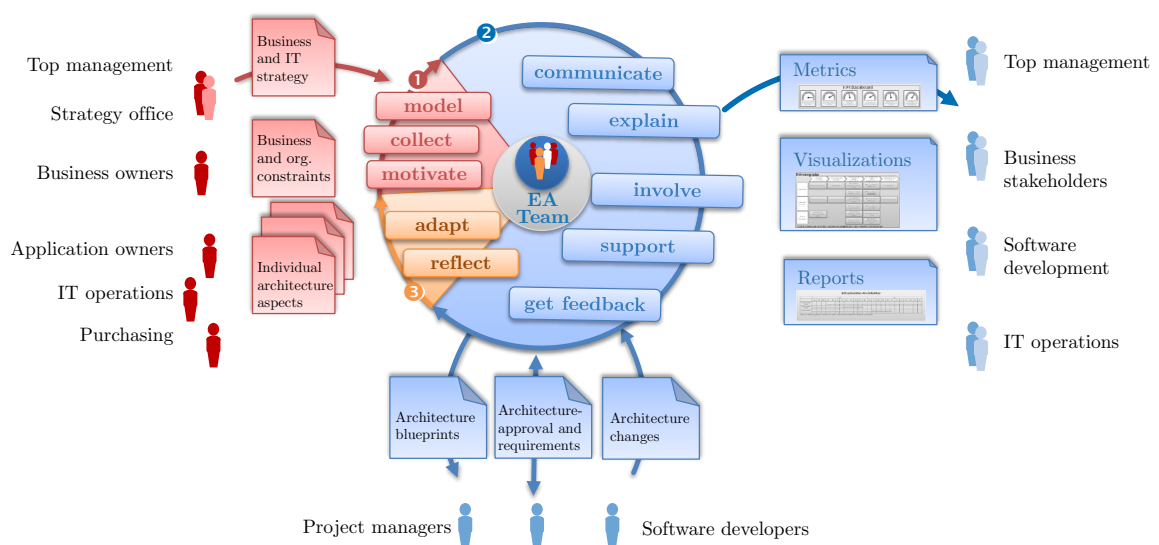


Figure 2.6: EA management as an iterative management discipline (based on [RZM14, p.12])

In the first phase (❶ in Figure 2.6), EA management starts by motivating an EA endeavor. Especially in the first iteration, value-generation does not take place during this preliminary phase. However, for a successful EA management initiative, it is of utmost importance to get the stakeholder buy-in, i.e. convince stakeholders of the meaningfulness of EA management and long-term benefits for the entire organization. Further, top management support is considered essential for successful EA management endeavors (cf. [YP13]). In [HSR⁺13], we provide empirical evidence of the impact of top management support on EA management.

- **Motivate:** At all times, the consideration of social aspects for EA management is of major importance. In line with Ahlemann et al. [ASM⁺12, p.138], we find that top management support is a crucial success factor for an EA management initiative; without top management support an EA management endeavor is likely to fail. Existing management functions that coordinate efforts across business units, e.g. the strategy office, may provide a business and/or IT strategy that can serve as a starting point for EA management, focusing on aligning business and IT strategy. Equally important, however, is the stakeholder buy-in and proactive stakeholder engagement. It is of utmost importance not to underestimate the role of stakeholders for a successful EA management initiative. Important stakeholders include but are not limited to business, project managers, and application owners; but also technical staff such as IT operations (cf. Hanschke [Ha10, p.99ff]).
- **Collect:** Next to motivating EA management, the EA team collects information that serves as a decision base later on. The collected information is formalized by developing models of an EA that reflect the reality. In the first iteration, any initial information should be gathered. This information can build a profound basis for a conceptualization. In subsequent iterations of this phase, a more focused way for data gathering should be chosen. Challenges that arise during this phase are detailed in Chapter 3. Usually, an EA model conforms to an EA metamodel that represents a conceptualization of the different entities, their attributes, and their relationships to each other.
- **Model:** Hence, an EA model (mental or explicit) is conceptualized or the existing conceptualization is extended or adapted to reflect the current circumstances within an enterprise. Ideally, each aspect that addresses a concern of a stakeholder is reflected by the conceptualization of the EA. An anti-pattern reported by Ambler et al. [ANV05, p.194] is the ‘*ivory tower architecture*’: the EA looks good on paper commonly fully blown conceptualized but does not reflect the reality. Enterprise architects must understand their stakeholders and incorporate their feedback in the EA model and respective conceptualization. Moreover, the focus should be put on the most important aspects which generate value.

In the second phase (❷ in Figure 2.6), developed models and concepts are communicated and used to explain decisions and the long-term benefit and further course of EA transformation projects. Backed by top management, the EA team nurtures an organizational culture that embraces the EA initiative. In this phase, the EA team should show the turnover for each individual stakeholder. While EA management is meant to realize mid-term to long-term goals, it is important to justify intermediate costs for EA management. In this step the

agile principle “Individuals and Interactions over Processes and Tools” should guide an EA management initiative (cf. [HRS⁺14]).

- **Communicate:** In EA management, the communication of results is facilitated using metrics and calculated KPIs as well as visualizations and reports. The concrete representation form depends on the stakeholder (cf. Section 2.1.2). Top managers for instance make decisions on highly aggregated KPIs whereas visualizations displaying interconnectedness and eventually causal dependencies are valuable for business stakeholders as well as for software development and architects. Since collecting required information commonly is regarded cost-intensive, a small number of visualizations and KPIs, which generate most value to the organization should be maintained, instead of wasting efforts with the time-consuming composition of outdated and defective reports that seek to explain and forecast everything. Other important factors are to speak the language of the EA Stakeholders and to cultivate relationships to members of other management functions, e.g. PPM or change management (see *involve*).

- **Explain:** The EA model as well as its usefulness for the organization as a whole and for individual members of the organization should be communicated and explained explicitly.

During this activity, the EA team explains not only the benefits of EA management and the need to maintain an EA model but further provides rationale of architectural decisions (cf. Section 2.1.2). These can be incorporated for example in visual representation of planned states.

- **Involve:** It is important to involve stakeholders at all levels, i.e. top management, business stakeholders, software developers, IT operations, solution architects, project managers, etc. The EA team might support stakeholders in solving their individual challenges such that the efforts for the information collection to keep the EA model up-to-date are justified. This often can be accomplished by providing necessary information and highlighting the common good for the company that EA management provides (see *communicate*). The quality of the information base that is referred to as EA model is crucial for the trust in an EA management initiative. One way to ensure a sustainable model quality is through incentives and integration in organizational processes, e.g. PPM and change management (cf. Section 2.1.3 and Section 3.2.1).
- **Support:** In this phase it is important to avoid negative effects of tools on team collaboration and EA models. In line with Ambler et al. [ANV05] in [HSR⁺13], we empirically confirmed the issue of an over-sized EA information model. Hence it is important to avoid ‘waste’ in terms of gathering information that in the end is not used. Rather we advocate to focus strictly on the actual information demand of top stakeholders.
- **Get feedback:** A common step to iterative process designs is to gather feedback on the method and steps undertaken to feed an administrative method that seeks to improve the process. Typically, EA Stakeholders are asked what went good and what went wrong, e.g. during the execution of their software development project. This way, feedback concerning the EA management process is gathered directly and indirectly.

In the third phase (③ in Figure 2.6), EA management should reflect their practices and outcomes. EA management still is a young discipline; researchers and practitioners should synthesize building blocks and patterns of best-practices and constantly re-evaluate applied methods (cf. [BMM⁺11a]).

- **Reflect:** This activity serves not only to analyze and critically reflect the outcomes of each iteration but also to analyze and reflect stakeholder feedback and engagement. This activity seeks to find root causes of deviations between planned and actual achievements. Given a new basis of information, the EA team may propose changes to processes.
- **Adapt:** This final activity serves to adapt the EA management function as well as organizational processes. These may have to be adjusted in response to the produced results or the feedback gathered in the previous phase. An example for an adaptation of the organizational process is a new policy that require approval of architectural changes by the EA team, which in turn might have a finite (and shared) set or requirements (cf. principles and standards in Section 2.1.1) for the EA or individual solution architectures.

A critical reflection of the EA management team’s behavior as well as delivered artifacts requires constant feedback. The EA management team has to demand feedback and adapt their models and methods if needed. This especially emphasizes the human and social aspect of EA management. A successful EA management initiative relies on continuous collaboration between the EA management team, their stakeholders as well as top management support.

In [BMM⁺11a] we propose to proceed iteratively and incrementally such that a cycle (phase 1-3 in Figure 2.6) lasts no longer than 12 months. Meanwhile, EA management initiatives should take changes in business and technology into consideration. Ahlemann [ASM⁺12, p. 231] proposes that an iteration should not take longer than three to nine months. Although many of our empirical investigation confirm that TOGAF is the prevalent EA management framework, we conclude that, as of today, there is no established standard we can refer to and diagnose that empirical evidence for the ideal time-frame for an iteration is missing. Similar to the conceptualization of an EA model this may depend on organization-specific factors.

Note that this process is meant to give an overview of our current understanding of EA management; we are aware of many other factors that are important for EA management. Especially Aier [Ai13] and Hauder et al. [HSR⁺13] outline cultural and organizational specifics that seem to have an impact on EA management.

2.1.5 Summary of EA and EA Management

In this section, we established a general understanding of EA management. Therefore, we looked at EA management from a rather static perspective. Thereafter, we emphasized involved elements by referring to the ISO 42010 standard. Finally, we gave a more dynamic perspective on EA management and described a typical EA management process. Thereby,

we pointed out the social component that is of utmost importance for a successful EA management initiative. In the remainder of the thesis, we build on these groundwork: Our approach adds information from different modeling communities incrementally and involves stakeholders during the resolution of model conflicts.

Although EA management depends on a sound information based stored in an EA model, current frameworks for EA management, e.g. TOGAF [Th11] or Zachman [Za87], do not provide any guidance for information procurement [FAB⁺11b]. This holds true for the lean approach to EA management we presented above. In the remainder of this thesis, we establish that a software-supported process design can facilitate some of the important aspects of EA management.

2.2 Modeling and EA Modeling

The description of an EA is a means to derive planned states that describe transformations leading towards an envisioned target state. Thereby, the description of an EA is an explicit artifact (cf. Section 2.1.2). This description is formalized in a model. In this section, we briefly revisit the fundamental features of a model to build a common understanding of models. Further, the different levels of abstraction commonly employed as a frame of reference are sketched.

2.2.1 Fundamental Properties of a Model

According to Stachowiak [St73, pp. 131-133] there are particular features a model always exhibits⁵. Those features are:

Mapping feature: Models always refer to an original—a so-called *universe of discourse*. Models are mappings or representations that serve as surrogates of objects in the physical world or of artificial or mental originals.

Reduction feature: Models commonly do not include every attribute of an original, but rather limit the scope relevant to their respective model creators and/or other stakeholders. The art of reducing the model to such a purposeful scope is called *abstraction*.

Pragmatism feature: Models are not uniquely assigned to their originals in the real-world. They fulfill a replacement function

- for a particular subject (human or artificial receiver),
- within particular time intervals, and
- restricted to particular mental or actual operations, i.e. models serve a special purpose and are a means to an end.

⁵We refer the interested reader to Thomas [Th05]. He presents a comprehensive review of different notions of models and an extensive discussion on the topic.

For the remainder of the thesis and with respect to the domain of EA management, we define:

DEFINITION 2.1: Model

Essential properties of a model are that

- a model is built by a *model creator* at a particular time,
- a model abstracts from reality, i.e. physical or mental originals in the *real-world*,
- a model serves model users (stakeholders) for a specific *purpose*.

■

A model can be conceptualized by defining properties of specific concepts in the real-world that are of interest for the modeler. From an object-oriented (OO) perspective, each physical object is an object that conforms to an entity. Such an entity has attributes. A special kind of attribute refers to other objects. We call these relationships. We refer to such a conceptualization of a model as metamodel.

DEFINITION 2.2: Metamodel[†]

A metamodel is a conceptualization of a class of models. It is an explicit conceptual description of entities, attributes, and their relationships to each other. This description forms a language that determines how to specify permissible models (cf. [Fr11, p. 39]).

[†]Synonym(s): information model [Le99], schema, conceptual schema [HM10, p. 27]

■

In Section 2.1.1, we already established the role of modeling languages that are employed to formalize a model. We also established an intuitive understanding of the term metamodel. In the following, we investigate the relationship between models and metamodels more closely.

2.2.2 Meta-levels of Models

Figure 2.7 illustrates the different levels models (cf. [OMG11a, pp. 17–20]). Each level conforms to the next upper level, i.e. is an instance thereof.

From bottom-up, we begin with the reality. Depending on the community, whether modeling or database management system (DBMS), two prevalent terms exist to describe the reality. Each of the terms is used exclusively by only one community. The modeling community speaks of a model whereas the DBMS speaks of *data* that is physically stored within a database managed by a DBMS. Both capture the real-world, are created by a particular person (modeler), and serve a purpose, i.e. both terms conform to our notion of a model (Definition 2.1).

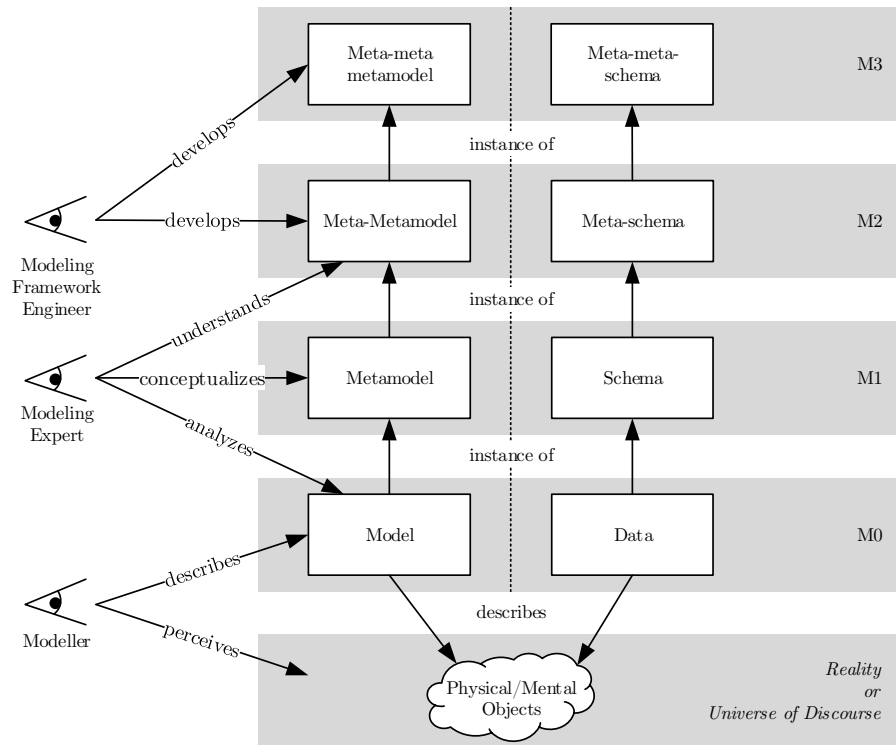


Figure 2.7: Relationship of model, metamodel, meta-metamodel, and meta-meta-metamodel

This model—or data—can again be conceptualized as we established in Preliminary Definition 2.1. Thus, a model is an instance of a metamodel whereas the DBMS community⁶ refers to the conceptualization of data as the *schema* or database schema. This modeling process demands considerable analytical skills. A *modeling expert* has to

- understand the model and perhaps also the circumstances in the real-world described by the model,
- conceptualize the model in a metamodel, and, thus,
- understand the metamodel offered by the respective meta modeling facilities employed (cf. Section 2.1.1).

Before we proceed with the discussion, we provide Example 2.1.



EXAMPLE 2.1: Developing a schema for a database

The modeling expert—a Database Administrator (DBA)—gets the initial data (model) on a hand-written piece of paper. He is trained expert for a particular DBMS (meta modeling facilities) and, thus, develops a schema (metamodel). It is not necessary that the user of a system understands the schema. Still, the

⁶In this thesis, we view DBMS from a user and Database Administrator (DBA) perspective; we refer to Date [Da04, p. 34ff] for the classical three level architecture of database systems.

user can describe and manage information about entities in the real-world with the corresponding information system that accesses the database and provides read and write access.



A meta-metamodel is (commonly) a means to realize a modeling language. The meta-metamodel of such a language typically conforms to a meta-meta-metamodel. A prominent example for meta-metamodel is the Unified Modeling Language (UML) which conforms to a meta-meta-metamodel, namely the Meta Object Facility (MOF) as the self-contained foundation for UML (cf. [OMG11a, p.14]). We discuss these rather distinct notion between the different layers in Section 5.1.2.8 and meanwhile refer the interested reader to Atkinson and Kühne [AK01b, AK03] for an interesting discussion on the topic. To conclude our description of Figure 2.7, a meta-schema is what a DBMS system commonly implements whereas—to our best knowledge—following this logic for the DBMS community no such thing as a meta-meta-schema exists. This would be one possible solution to build a common denominator among different DBMS implementations. Harmonization efforts of different DBMS systems are further discussed in Section 2.4 where we also introduce the different schemas that interact within a federated database system (FDBS).

2.3 Model Evolution and Merging

Next, we draw a parallel between software merging and model merging. Software can be considered a model (source code) that conforms to a metamodel (programming language). The model evolution and merging community originates from source code merging. For this purpose, line-based approaches are utilized to merge changes on source code. Many publications, cf. e.g. [Me02], underpin that line-based approaches, e.g. [HM76, He78, My86], are not sufficient for software or source code merging (cf. also [Ti84]). Respective solutions proposed to solve related issues, e.g. [Li04, LKT06, TBW⁺07], are very similarly to model merging approaches, e.g. [KWN05, BP08]. In particular, programming language independent approaches, e.g. [Me99], are faced with similar problems outlined by the model merging community, e.g. [BSW⁺09, KPR13, TEL⁺10, WLS⁺12]. Mens [Me02] outlines structural merging of source code and exemplifies related issues using UML class diagrams, i.e. models (cf. Definition 2.1).

Rutle et al. [RRL⁺09] motivate model merging from another angle, namely model-driven engineering (MDE). MDE is an often applied methodology in software engineering. Models as well as their change over time, i.e. their evolution, are essential for MDE. Thus, many authors, e.g. [Ba08, DMJ⁺07, KHS09, NMB⁺05, OK02, RW98], motivate the need to merge not only source code but also models in a repository, describe arising issues, e.g. [LW13, WL13], propose solutions, e.g. [Ba08], and carry out empirical evaluations of different merge approaches, e.g. [KHS09].

In his extensive state-of-the-art survey Mens [Me02] revisits literature on the topic of software merging and identifies multiple dimensions to compare the different approaches. While Mens puts emphasis on software merging, Koegel et al. [KHL⁺10], Kolovos et al. [KDRP⁺09],

and Rose et al. [RHW⁺10] concentrate on models. Literature often compares model merge approaches based on a particular dimension, e.g. [KDRP⁺09, KHL⁺10]. Focus is often put on matching elements [KDRP⁺09, SE08, LHS⁺06], mapping elements [RB01a, RB01b, BM07, VIR10], detecting conflicts [LW13, TEL⁺10, KHS09, As94, Ed97] and resolving them [Br12, KHS09, CDRP08, SP94], or runtime behavior of employed algorithms [RC13], i.e. computational complexity.

In the following, we outline core concepts of this community to foster an understanding of important terms of the software and model merging communities.

2.3.1 Two-way, Three-way, and N-way Merging

The first step towards a merge of two versions of source code is to calculate their differences [Me02]. Mens [Me02] distinguished between two general merge or differencing approaches:

- **Two-way differencing:** A *two-way differencing* approach compares two versions of an artifact without considering their origin.
- **Three-way differencing:** A *three-way differencing* approach utilizes knowledge about the (common) origin of both versions to be merged to calculate differences. When multiple versions originate from the same version, i.e. they are branches thereof, we call this a *common origin*.

Two-way differencing has its drawbacks. Namely, with a two-way differencing it is often hard or not at all possible to tell whether differences are caused by adding, removing, or updating information from a previous version. As a reaction to these shortcomings, today's source code management (SCM) tools, e.g. Subversion (SVN) [Ap14], Git [Ch08, Ch09], or Mercurial (HG) [Me14], use three-way differencing approaches by default. Similar to these approaches, we consider the common origin (cf. Section 5.2.5). Example 2.2 shows a two-way as well as a three-way approach to illustrate their differences more clearly.



EXAMPLE 2.2: Two-way and three-way merge of text files

We can use the Unix utilities *diff* [HM76, HS77] and *diff3* to calculate differences⁷ of a small piece of source code written in the Java programming language illustrated in different versions (cf. Figure 2.8). The *diff* utility compares version 1a with version 1b without knowing the common origin version 1. Based on this two-way difference calculation (cf. result of *diff* in Figure 2.8), it cannot be decided whether lines have been added, some lines have been deleted, or are just updated.

⁷We refer the interested reader to [HVT98] for an overview of delta algorithms and their runtime behavior.

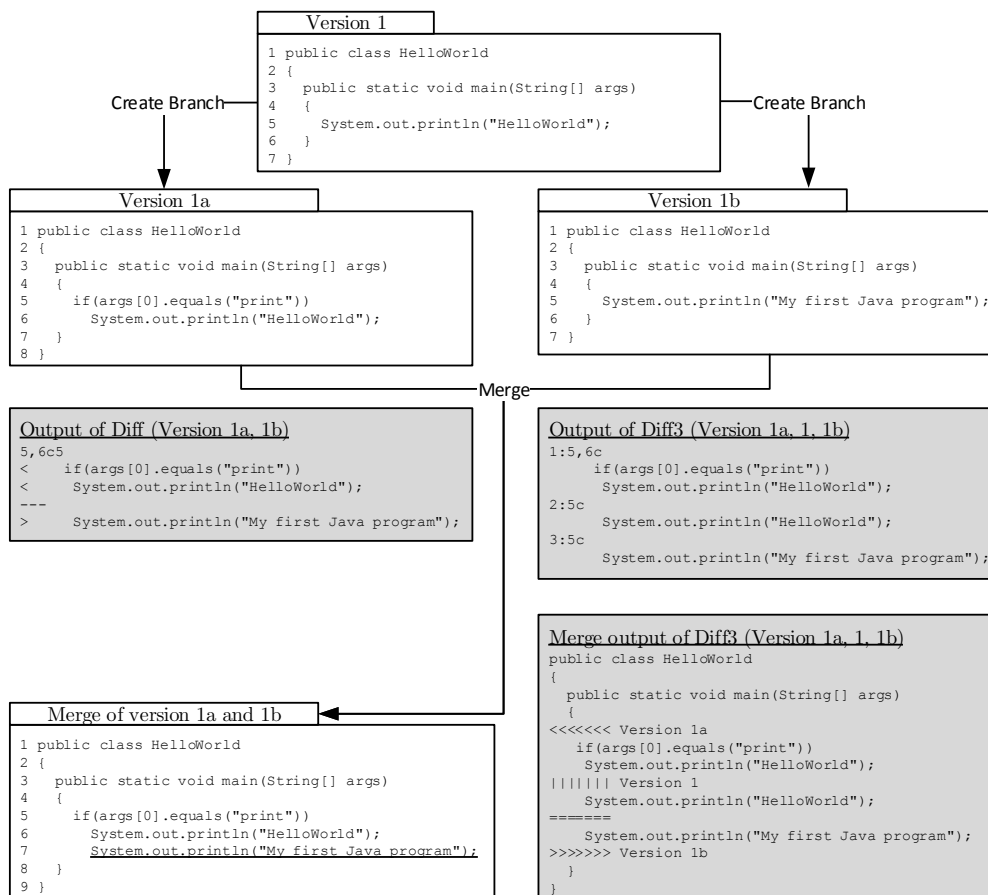


Figure 2.8: Merge of source code using the Unix diff and diff3 utilities

In contrast, the diff3 utility also considers the common origin. The subsequently applied merge strategy can take into account that only the value of the string in line 5 has changed. Further, the if statement (line 5 of version 1a) is added to the merged version.

To resolve model conflicts, it is often beneficial to merge multiple versions at once to have all the intended changes at hand [RC13]. Rubin et al. [RC13] provide a formal definition of n -way model merging [RC13]. In their paper, Rubin et al. [RC13] use weights on model elements to merge n models and reduce the matching problem to the NP-hard problem of ‘*weighted set packing*’ [AH98]. While we agree with the general problem that n versions of a model or subsets thereof must be considered to find or calculate an ‘optimal’ merge result, for EA model maintenance, this is a highly manual task. The merge and conflict resolution in an EA model is influenced by many factors which undoubtedly cannot be reduced to weights in a fully automated algorithm. Thus, the present thesis takes a different perspective (cf. Section 5.2). Definition 2.3 summarizes our notion of an n -way merge (cf. [KR14]).

DEFINITION 2.3: N-way merge

An n-way merge approach utilizes knowledge about the origin of n versions to be merged to calculate differences provided that $2 < n < \infty$. When multiple versions of a model element $\mathcal{E}_{2..n}$ originate from the same version \mathcal{E}_1 , i.e. are branches of \mathcal{E}_1 , \mathcal{E}_1 is called common *ancestor*, *origin*, or *provenance*. ■

In Chapter 5, we build on the foundations outlined above. Thereby, two-way and three-way differencing as well as our notion of an n-way merge are of special interest. We provide an algorithm that calculates differences between models (cf. Section 5.2.4) and an algorithm that realizes an n-way merge (cf. Section 5.2.5) with integrated models that build a federation (cf. Section 5.2.1).

2.3.2 Textual, Syntactic, Semantic, Structural Merging

The result of the merge in Example 2.2 shows that also syntactical merge conflicts may arise in the event of source code merging. In the illustrated case, it remains unclear whether both `System.out` statements should be included or not (cf. line 6 and 7 in merged version of Figure 2.8). When both statements are included, the brackets of the if statement could be missing. On the one hand, this depends strongly on the intended behavior of the program. On the other hand, it also depends on the syntax of the programming language. However, text-based, i.e. line-based, approaches cannot resolve such conflicts.

To address shortcomings of text-based approaches, Buffenbarger proposes syntactic software merging [Bu95]. Key to this approach is to apply merge strategies not to text but to the structure of the source code resulting from the parsing process, i.e. an Abstract Syntax Tree (AST). Mens [Me02] distinguishes syntactic merge approaches by their underlying data structure. This can be either tree-based or graph-based. Graph-based approaches outperform tree-based ones when it comes to reflecting cross-references as this is the case for instance for function or method calls. For instance Rho and Wu [RW98], Taentzer et al. [TEL⁺10], and Mens [Me99] use attributed graphs as an internal representation of software artifacts.

Another approach is ‘*semantic merging*’. Mens distinguishes between ‘*static semantic conflict*’ and ‘*behavioral conflict*’. The former refers to syntactical errors that would raise exceptions by the compiler, e.g. an undeclared variable; the latter describes conflicts in the runtime behavior of the executed code. Semantic merging is strongly related to source code—be it static or runtime characteristics—and thus respective approaches are regarded out of scope for the present thesis.

Instead, another kind of approach is of interest—structured merging [Me02]. These approaches are motivated by demands to increase maintainability of source code while preserving its functionality and behavior, i.e. software refactoring as advocated by Fowler [Fo99], Griswold [Gr91] and Opdyke [Op92]. Related conflicts are coined by Mens as ‘*structural merge conflicts*’ [Me02]. Mens notes that “existing merge tools and techniques provide no support whatsoever for detecting structural merge conflicts. The main reason is that the information required to detect these conflicts is usually implicit and cannot be inferred

from the source code only” [Me02]. This holds true for EA model maintenance; especially when comparing different EA metamodels, the (original) intention of the modeler is often unclear or remains implicit.

2.3.3 State-based, Change-based, and Operation-based Merging

An orthogonal dimension is mentioned by Koegel et al. [KHS09] as well as Mens [Me02]; they distinguish between:

- **state-based** approaches compare two different model versions, i.e. a version and its successor after changes have been made [CW98]. This post-mortem analysis is referred to as *differencing* [Me02].
- **change-based** approaches track changes while they occur such that there is no need for differencing since changes are stored in a repository explicitly [KHL⁺10].
- **operation-based** approaches are specialized change-based approaches [LO92] defining transformations in such a way that a state of a model is turned into the state of its successor [CW98].

To further distinguish these approaches, we briefly discuss their advantages and disadvantages. Change-based approaches often are implemented as operation-based approach. Thus, we focus primarily on state-based and operation-based approaches in the following. A major advantage of state-based approaches is their tool independence [KHL⁺10], i.e. one does not need a prescribed tool to manipulate a model. The operations performed on a state-based approach are immediately applied to the model, i.e. operations alter the model’s state. In literature [Me02, RL07], change-based and operation-based approaches are considered more powerful than state-based approaches; state-based approaches exhibit several disadvantages:

- **No temporal order of changes:** Temporal order cannot be derived anymore. Temporal order can be important for understanding changes or for detecting conflicts, e.g. dirty updates [LO92].
- **Groups of changes to composite changes are lost:** Atomic changes are often hard to understand for users; they are too technical and are related to fine-grained operations of the modeling framework. In particular Kehrer et al. [KKO⁺12] addresses this issue for EMF-based solutions. The authors investigate the semantics of multiple primitive operations. Xing et al. also analyze and group changes to detect refactorings [XS06].
- **Computational complexity:** Especially for large models the required computational effort is high [LKT06, TBW⁺07]. Differencing of models is commonly regarded expensive in terms of computation time. On the one hand computational time can be attributed to the complexity of the algorithms; on the other hand, the size of the models is also an important factor.

In particular operation-based approaches have several advantages for conflict detection and merging [DMJ⁺07, LO92, Me02], repository mining [BMM⁺08], and coupled evolution [HBJ09, Wa07]. Against this background, we employ an operation-based approach for the merge of different models in our solution design (cf. Section 5.2.5.2). Moreover, we employ a state-based comparison (cf. Section 5.2.4) to visualize differences between the latest revisions of models. Although thousands of elements are captured in an EA model, an EA model is considerably smaller than a software model—thus, the present thesis does not focus on reducing complexity of differencing and merging algorithms.

2.3.4 Static Identity-based, Signature-based, and Similarity-based Matching

Kehrer et al. [KKP⁺12] and Kolovos et al. [KDRP⁺09] distinguish between model-type-specific and generic approaches. For the present thesis, we focus on generic approaches, i.e. approaches supporting an arbitrary metamodel. Kehrer et al. [KKP⁺12] and Kolovos et al. [KDRP⁺09] further categorize generic approaches in ‘*static identity-based*’, ‘*signature-based*’, and ‘*similarity-based*’ approaches.

Static identity-based approaches use a non-volatile unique identifier (UID) for each model element [KDRP⁺09]. This persistent UID is assigned to a model element upon creation. The core idea of static identity-based approaches, e.g. [AP03, FGC⁺06], is to match models and respective elements based on corresponding identities. Kolovos details two main advantages. First, no configuration by end-users is required and second, utilizing UIDs to match model elements is particularly fast. However, neither can it be applied to models built independently nor to modeling frameworks that do not support the maintenance of UIDs [KDRP⁺09]. Further drawbacks of this approach are discussed in [SK07].

Signature-based Reddy et al. [RFG⁺05] propose signature-based matching to overcome limitations of static identity-based approaches. Instead of using static UIDs, the signature-based approaches calculate the identity of model elements dynamically. These calculations are specified as functions over model elements values employing a model querying language. As this calculation is done dynamically, the signature-based approaches overcome one of the drawbacks of static identity-based approaches, i.e. they can be applied to models build independently from each other. On the other hand, with signature-based approaches developers or users have to specify the identity function of different model elements [KDRP⁺09]. Kehrer et al. report that “the signature is typically a hash value which comprises one or many conceptual properties of model elements, including their type or technical properties such as persistent identifiers of model elements” [KKP⁺12]. That means in practice, static identity-based and signature-based approaches are also mixed.

Similarity-based Both approaches, static identity-based and signature-based, calculate a boolean value, i.e. true or false, for a match of model elements. In contrast, similarity-based approaches seek to match the most similar model elements [KKP⁺12]. Thereto, a configuration defines how similarity matches are calculated since not all model

elements are equally important, e.g. classes with similar names are more likely to be a match than classes with similar values [KDRP⁺09]. Thus, the configuration specifies the relative weight of model elements. For each type this configuration includes relevant properties and a respective function that computes the similarity of two property values. Additionally, another function computes the total similarity of two model elements [KWN05, TBW⁺07]. Some approaches, e.g. [TEF13, RV08], utilize one standard function for any property [KKP⁺12]. Mens [Me02] compares identity-based with similarity-based matching and concludes that latter delivers more accurate results. However, Mens also notes that the fine-tuning of weights is a challenging and long-lasting trial-and-error process.

In the remainder of the thesis, we provide a solution design that—to some extent—combines static identity-based and signature-based approaches (cf. Section 5.2.4.3).

2.3.5 Tentative Merge Results

In their ‘*fundamental approach to model versioning*’, Taentzer et al. [TEL⁺12] propose an interesting approach that inspired our work. The authors consider models as evolving artifacts, which especially holds true for MDE. They also draw an analogy to text-based version control system (VCS) solutions and propose a graph-based approach that takes model structures and their evolution into account. Model changes are thereby viewed as graph modifications consisting of primitive delete and insert actions. Their approach to conflict-detection is two-fold, i.e. Taentzer et al. check graph modifications for operation-based conflicts and employ the resulting (merged) graph for state-based conflict detection. For the latter, they temporarily resolve operation-based conflicts preferring inserts over deletes. This way, they keep as much information as possible. They call this intermediate model a ‘*tentative merge result*’. This merge result builds the basis for “manual conflict resolution as well as for the application of repair actions” [TEL⁺12]. In her PhD thesis, Brosch [Br12, ch. 6.3.2] also employs a tentative merge.

In Section 5.2.5.4, we apply the general idea of such an ‘*tentative merge result*’ to be able to quickly revert changes and preserve any information. Further, we propose an optimistic conflict resolution strategy that preserves as much information as possible (cf. Section 5.2.8.2).

2.3.6 Conflict Detection

There are different hard-coded and more flexible ways to detect a conflict [Me02]. In his comprehensive state-of-the-art article, Mens [Me02] reports on:

- **Merge matrices:** This category subsumes techniques to detect conflicts which employ a ‘*merge matrix*’ or ‘*conflict table*’ to manage operations. Thereby, conflict detection is often reduced to a pair-wise lookup in a lookup table (LUT), i.e. pairs of operations which may lead to a conflict are used to perform the lookup whereas the result defines how the system reacts. Examples of such approaches are Feather [Fe89], Steyaert

et al. [SLM⁺96], Munson and Dewan [MD94], Mens [Me99], Lippe et al. [LO92], and Kirschner et al. [KR14].

- **Conflict sets:** Approaches of this category group together potential conflicting situations, i.e. allow to specify several patterns of operations that—potentially—trigger a conflict. Edwards [Ed97] proposes such an approach. Moreover, Edwards also proposes to tolerate conflicts within the system.
- **Semantic conflict detection:** For the sake of completeness, the final category uses either implicit or explicit semantics of programs to detect conflicts. We regard this category of little interest for the present thesis and refer the interested reader to the excellent study by Mens [Me02]. He summarizes further approaches and outlines some of their shortcomings.

The first two approaches had considerable impact on our designs that led toward this thesis. In [RHM13b], we propose an approach for manually specifying an explicit conflict set, i.e. the conflict set does not serve to detect a conflict but is used for the resolution of manually detected conflicts by involving users. In [KR14] we propose a configurable approach and concrete conflict resolution strategies (cf. Section 5.2.8). This approach is similar to merge matrices (cf. Section 2.3.6).

2.3.7 Conflict Avoidance, Tolerance, and Resolution

Conflict resolution and conflict tolerance is perhaps the most significant difference between software and model merging. Especially when the model is used to capture information and is not used for execution, conflicts are perceived differently. In contrast to MDE, an EA model is not meant for execution and especially syntactical flaws in specific attributes, e.g. type constraint violations, commonly have little impact.

Based on the works of Mens [Me02] and Edwards [Ed97] and in line with Brosch et al. [BLS⁺10] we distinguish between three different approaches:

- **Conflict avoidance** seeks to avoid conflicts entirely. Brosch et al. provide two examples [BLS⁺10]: by realizing a pessimistic strategy, e.g. with explicit locking mechanisms or by synchronous modeling activities.
- **Immediate conflict resolution** tries to keep only consistent versions of a model. Brosch et al. [BLS⁺10] note that in traditional VCSs, the developer who checks in last bears the responsibility to produce a working, i.e. consistent, piece of software.
- **Conflicts tolerance** allows (temporary) inconsistencies. Brosch et al. [BLS⁺10] use a model (with tentative merge results) as a basis to resolve inconsistencies and apply repair actions.

Especially the last strategy assumes that by ignoring inconsistencies during a merge gives modeling experts the opportunity to see all conflicts. In the approach of Wieland et al. [WLS⁺12] respective conflicts are preserved as annotations on a model element such that modeling intentions can be discussed collaboratively (cf. Section 5.3.2).

Another approach from the software merging community is proposed by Asklund [As94]. Thereby, a fine-grained revision control is employed to keep persons aware of each others changes. The authors expect to minimize conflicting changes which in the end makes a merge simpler. Perry et al. [PSV01] even observe that it is easier to merge small changes on a frequent basis. The authors note that a merge of larger changes leads to very costly overhead and increased coordination effort. Policy-based approach give guidelines for organizational regulations. For instance, Bruegge and Dutoit [BD99] propose to develop software in one main branch; respective changes in this branch should be restricted to bug fixes. Features should be developed in separate branches. At the same time, the total number of branches should be as small as possible.

In Section 5.2.1 we detail the role of branches for our approach. Additionally, conflict tolerance and a considerable degree of freedom during modeling and meta modeling within and across communities is central to our approach (cf. Section 5.2.8.2). We propose a fine-grained access control (cf. Section 5.1.2) which is also used to notify responsible persons within an organization. Observations made by Perry et al. could be transferred to EA models as it is more likely to succeed in merging small changes on a frequent basis with an EA model. However, there are exceptions: For instance, in the course of a new major release of a commercial off-the-shelf (COTS) product, the model and metamodel could be migrated such that considerable structural changes occur in a repository of a modeling community.

2.3.8 Collaboration

In the context of structural merging (cf. [Me02]), Mens also mentions the need of user intervention to resolve conflicts—an idea that guides our design (cf. Chapter 5). Since Mens' study [Me02], several approaches coping with models and related conflicts have been developed, e.g. [GKL⁺13, Br12, HRP⁺13b, Wi11].

Traditionally, a single developer is in charge of the merge process [Me02, BLS⁺10]. Wieland et al. [WLS⁺12] propose a different approach. First, they merge models in a conflict tolerant manner annotating a model that incorporates tentative merge results. Thereafter, this model serves as a basis for collaborative activities that seek to resolve model conflicts. The annotations capture not yet applied, concurrent, modeling operations.

In the present thesis, we detail a collaborative approach as proposed for MDE by Wieland (cf. [WLS⁺12]). Thereby, we extend the existing notion of conflicts (cf. Section 4.2.7 and 5.2.5) and provide means that intends to foster collaboration and conflict resolution (cf. Section 5.2 and [RHM13b, HRP⁺13b]).

2.3.9 Persistent Versions

SCM systems rely on delta algorithms [HVT98] to reduce storage space. Applying delta algorithms commonly means less input/output (I/O), since only deltas must be read from storage. According to Mens [Me02], the I/O time saved is even higher than the required time to regenerate entire revisions. Assuming Moore's Law [Mo65, Sc97] holds true, delta algorithms will remain an important dimension to consider when designing a

system capturing revisions of artifacts. For traditional SCM we distinguish between the following approaches:

- **persistent deep-copy** refers to approaches that store an entire version for each change on a revision without applying any delta algorithm.
- **delta-forward** means that the original version is stored and for changes only the deltas to the current revision are stored.
- **delta-backward** describes approaches that store latest revision and for changes only the deltas to the original version are stored.

Mens states that two-way merge approaches commonly use a ‘*symmetric delta*’ calculation (see [CW98, p. 239]) to compute the differences between two versions [Me02]. For three-way, operation-based merge approaches, a ‘*directed delta*’ approach calculates the sequence of operations between two versions.

In our approach, we apply a variation of the delta-backward approach. We store operations that incorporate information about the old revision and new revision. However, we persist the latest version which clearly qualifies for the delta-backward approach (cf. Section 5.2.5.2).

2.3.10 Visualizations

Many authors, e.g. [Ke97, We08], argue that visual support for model evolution and merging is a necessity. Some even use hand-crafted visualizations to explain their concepts, e.g. [KDRP⁺09]. Edwards [Ed97] outlines the importance to represent conflicts that occur in a collaborative environment in a visual manner and gives some design hints how to represent conflicts. Since then, visual representations for differencing [KKO⁺12, OWK03, RM14, We08] and conflicts [BLS⁺10, Wi11] have been proposed.

In [KDRP⁺09], Kolovos et al. discuss the role of visualizations for model matching and differencing. The authors mention that the presentation of differences varies by scope and only some pieces of information might be highlighted. The concrete syntax which renders the abstract syntax thus may even vary between diagrammatic or textual notations. Consequently, the authors distinguish between the internal calculation, an abstract representation of differences, and concrete visual presentation. We already established that an EA model serves different stakeholders by addressing particular concerns and commonly views are utilized as a means to communicate a specific viewpoint of architectural descriptions (cf. Section 2.1.2). In line with Kolovos et al. and in order to separate models and views, we provide an abstract syntax for model differences in terms of a metamodel capable to store model and metamodel differences in Section 5.2.4.1 and a concrete syntax for model differences in terms of a multi-layered visualization in Section 5.3.1.

Kehrer et al. [KKO⁺12] describe the challenge of communicating technical model changes to end-users. These are often too fine grained. As a solution, the authors propose ‘*SiLift*’, an engine to semantically lift model changes such that end-users understand model changes more intuitively. ‘*SiLift*’ can adapt a differencing engine, e.g. Eclipse Modeling Framework

(EMF) Compare [TEF13] or SiDiff [KKP⁺12], detect user-level model operations, and presents the results as an UML diagram as an Eclipse plugin (see [KKO⁺12]). A similar presentation is used in [KKT11] to annotate an UML model with changes.

Ohst et al. [OWK03] propose to visualize differences of UML diagrams. The authors not only present their concepts but also provide insights for layout considerations of the diagrams and semantics of different interactions. Further, they sketch related challenges and present tool support for their solution.

In [RHM13a, RHM13b, RM14] we present preliminary work and evaluation results that build the foundation for the visual support described in the present thesis (cf. Section 5.3). We propose a concept that picks up the analogy to UML and the layering principle (see Lidwell et al. [LHB10, p.146]) which previously has been transferred to the domain of system cartography by Wittenburg [Wi07, p.83].

2.3.11 Technology Stack

According to Kelter et al. [KPR13] most approaches to model versioning, comparison, merging, and evolution are based on EMF. EMF with its corresponding modeling core meta modeling facility Ecore [SBP⁺09, ch.5] is considered the de facto standard for model-driven development [KPR13].

In the present thesis, we take a different perspective. In line with the considerations outlined above (cf. Section 2.3.5 and Section 2.3.7), we present a more flexible way to maintain a model and its metamodel. Instead of fat-clients developed with the EMF, we present a web-based solution which fosters collaboration (cf. Section 5.3 and Chapter 6).

2.3.12 Summary of Model Evolution & Merging

In the preceding sections, we outlined important concepts coined by the model evolution and merging community which influenced and guided our design decisions that are described throughout this thesis.

One can learn from this community as they seek to cope with similar problems, and propose innovative solution designs. However, there is a considerable difference in the nature of the models investigated by EA research and model evolution and merging research. In the remainder of the thesis, we argue that informal modeling for EA management is sufficient since models are a means to an end. Thus, they are used for knowledge management primarily to derive or plan states of an EA and justify decisions. The model evolution and merging community on the other hand copes with problems during software integration, particularly during MDE. The desired result thus differs considerably. However, we revisited some interesting approaches that proposed collaboration, tolerant merging, and models that incorporate tentative merge results. These ideas are also incorporated in our solution design described in Chapter 5.

Since a direct comparison of approaches is beyond the scope of the present thesis, we refer the interested reader to Altmanninger et al. [ASW09]. The authors give a comprehensive overview of different approaches used for comparison, conflict detection and resolution in

(software) model versioning. An even more recent comparison of the different communities and tool-driven approaches is provided by Brosch [Br12, p.27]. Particulars of model matching and differencing are investigated by Kolovos et al. in [KDRP⁺09].

2.4 Federated Database Systems

The idea of a FDBS dates back to 1979 initially proposed by Hammer and McLeod [HM79]. Based on this idea McLeod and Heimbigner [MH80] proposed an architecture for FDBSs. The authors call for a contemporary approach to database system architecture and advocate that this requires the complete integration of data into a single, centralized database. According to McLeod and Heimbigner multiple logical databases can be supported by a DBMS. However, techniques for relating these databases are generally ad hoc. In [HM85] Heimbigner and McLeod present a consolidated version of their approach for a coordinated information sharing and interchange of information. Thereby, they emphasize partial, controlled sharing among autonomous components, i.e. databases. Since then, a multitude of approaches for FDBSs has been proposed. Consolidating this knowledge, Sheth and Larson [SL90] present a reference architecture for FDBSs.

In line with Conrad [Co97, p. 44ff], Sheth and Larson [SL90] and Popfinger [Po07, p. 9], we revisit three orthogonal characteristics of a FDBS:

Distribution of the different components within a federation is the first characteristics of a FDBS. Classical reasons for distribution of multiple DBMSs are often improved response behavior or availability [RG03, ch. 22]; for FDBS however, the distribution often is due to preexisting DBMSs, i.e. typically DBMSs exist before the FDBS is built.

Heterogeneity between the different components as well as between components and federal entities is the second characteristic of a FDBS. Three general types of heterogeneity exist [SL90, SK93, Su01, NRS⁺99]:

Syntactical heterogeneity. When the same data in two databases is represented differently, e.g. in terms of its structure [HG01], we call this syntactical heterogeneity among DBMSs and respective databases. This includes technical heterogeneity arising through diverse file formats, access protocols, physical representation, query languages etc.

Logical heterogeneity. Conceptual schemas of the data could differ, i.e. same or similar data is represented in a different logical structure. In databases this could manifest in terms of table design for an entity including table decomposition and column names. Logical heterogeneity may originate from applying different data encoding to the same data; for instance using different measurement scales, e.g. °F or °C, to record temperature. Respective values can have the same semantics but different numbers through different encodings applied.

Semantic heterogeneity. Same or similar looking data has—perhaps in another context—a different meaning, i.e. concepts look like they may be related but in fact are different. “Semantic heterogeneity refers to differences or similarities

in the meaning of [...] data” [HG01]. Hakimpour and Geppert [HG01] provide two examples: 1) two model elements can have the same intended meaning, but different names (synonyms); 2) two model elements might be named identically, while their intended meanings are incompatible (homonyms).

With respect to DBMSs, Sheth and Larson [SL90] also describe differences in supported constraints and query languages of DBMSs.

Autonomy of the different components is the third characteristic of a FDBS. Ultimately, it is autonomy that underpins the difference to a centralized approach. In FDBSs, typically existing systems are integrated to a federation. An intact autonomy of components implies that organizational responsibilities for each component can remain unchanged.

These properties also apply for a federated EA model environment (cf. Preliminary Definition 1.1). For subsequent discussions, we view a schema as a metamodel and data as a model; data conforms to a schema similar than a model conforms to its metamodel (cf. Section 2.2.2).

2.4.1 Interacting Schemas in Federated Database Systems

Heimbigner and McLeod [HM85] describe an architecture for FDBSs in which independent DBMSs are united into a loosely coupled federation to share and exchange information. In this vein, a federation consists of multiple ‘*components*’, and a single ‘*federal dictionary*’. The federal dictionary maintains the topology of the federation and oversees the entry of new components, i.e. autonomous databases [HM85]. Each component in this federation controls its interactions with other components by means of an ‘*export schema*’ and an ‘*import schema*’. The former specifies the information that a component shares with other components, while the latter specifies the nonlocal information a component wishes to manipulate. Besides an export and import schema, each component has a ‘*private schema*’. The notions of private, export, and import schemes according to Heimbigner and McLeod are detailed in the following with respect to the reference architecture of Sheth and Larson [SL90]. Sheth and Larson introduce a ‘*five-level schema architecture for federated databases*’ that can be used to compare different approaches to FDBS design. In the following we revisit their core ideas briefly and provide the reader with a conceptual model illustrating the interaction between different schemas.

Local schemas are the metamodel of components stored at the components locally. Heimbigner and McLeod refer to the local schema as the ‘*private schema*’ of a component [HM85]. Further they note that major parts of the private schema describe information available to the component and, thus, correspond to database in a non-federated environment similar to common DBMSs. Although most of the information described in the private schema remains local to the component, some parts of the application data and transactions relevant to the participation in the federation are exported to other components. Heimbigner and McLeod describe three different categories of federation-specific information:

- meta information, e.g. name and network address of the component,
- operations for data manipulation, e.g. accessing a type, and
- import and export schemas.

Sheth and Larson [SL90] note that the local schema is defined in a component-specific way, i.e. in the native format of the respective DBMS. Hence, different local schemas may be expressed in different, possibly heterogeneous formats.

Component schemas are considered the translation of local schemas in a unified format, the Canonical or Common Data Model (CDM) of the FDBS. The CDM builds the foundation for the schema integration of heterogeneous components within the federation. Sheth and Larson give two reasons for defining component schemas using a CDM, i.e.

- component schemas describe divergent local schemas employing a single representation and
- component schemas can be equipped with additional semantics which are eventually missing in the local schema.

Export schemas specify a subset of the respective component schemas. Thereby, an export schema describes information a component is willing to share. In line with Heimbigner and McLeod [HM85], Sheth and Larson [SL90] note that a component may not share all information to the entire federation and its users. Consequently, an export schema may include information about desired access rights for particular types. These access rights can be enforced either by filtering accessible data or by limiting the available transactions that can be executed by a component, e.g. [HM85, VPZ88]. In the approach of Heimbigner and McLeod [HM85] similar to the private schema, the export schema embraces federation-specific information and application-specific information; both are derived from the respective parts of the private schema.

Federated schemas represent an integration of multiple export schemas. A concept similar to that of federated schema is represented by the terms ‘*import schema*’ Heimbigner and McLeod [HM85], ‘*global schema*’ [LR82, HG01], ‘*global conceptual schema*’ [LBE⁺82] unified schema, and enterprise schema. Except the term import schemas these concepts are usually used when there is only a single overarching schema in the FDBS. Although some systems use a separate schema to store information on data distribution [SL90], commonly this information is included in the federated schema. For instance, Heimbigner and McLeod [HM85] refer to the federated schema as the ‘*import schema*’ specifying information a component desires to use from other components. Their import schema contains both federation-specific and application-specific information.

External schemas define (updateable) views tailored to the specific needs of users and/or applications. Sheth and Larson [SL90] name the following reasons to use external schemas:

Customization: Federated schemas tend to become large, complex, and are relatively difficult to change. External schemas are a means to specify a subset of

information in a federated schema relevant to particular users. Especially in the event of frequently changing users' needs, the external schemas can be changed more readily than federated schemas.

Integrity constraints: In addition to integrity constraints specified in the different schemas, the external schema could also be equipped with additional integrity constraints.

Access control: In analogy to the Access Control List (ACL) in an export schema providing access control for data managed by the component, an ACL can also be applied to the external schema to specify access for data managed by the FDBS.

Note that an external schema is an optional schema and yet there could be multiple external schemas within a FDBS.

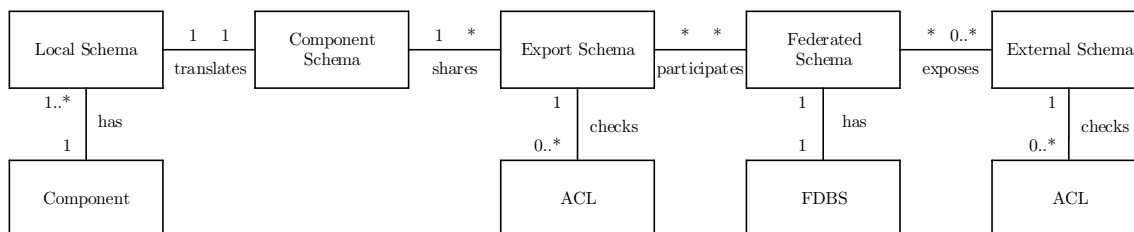


Figure 2.9: Component, FDBS, and interrelated schemas in the reference FDBS architecture of Sheth and Larson

Figure 2.9 gives a simplified conceptual UML class diagram that summarizes the relationships between the different schemas in the reference architecture for FDBSs introduced by Sheth and Larson [SL90]. Besides the concepts outlined above, a variety of alternative architectures exists, e.g. [BKL⁺99, HM85]. Thereby, not all schemas are required to build a FDBS. For instance, Heimbigner and McLeod [HM85] only rely on export, import and private schemas. Busse et al. [BKL⁺99] present a four-layer architecture for FDBSs that consists of a foundation layer, a wrapper layer, a mediation layer and a presentation layer [BKL⁺99, BKL00]. Essentially, Busse et al. summarize the component schema and export schema in one ‘*export component schema*’ that resides in the ‘*wrapper layer*’. Sheth and Larson [SL90] explain alternative architectures of FDBS. These alternatives come from redundancies between schemas and, thus, are a combination of the schemata outlined above. For further discussions and comparison of different alternative architectures to FDBSs, we refer to Conrad [Co97, p. 62ff].

2.4.2 Access Rights

The notion on access rights within a federation is described by Heimbigner and McLeod [HM85]. They assume that a component is willing to share certain types with every component, whereas it may also contain types that are shared only with a specified subset of the components in the federation. Thus, they place access controls on types and maps in the export schema; i.e. they equip the export schema with an Access Control List

(ACL) specifying which components may access a particular type and a list specifying which components have imported this type and, thus, are potentially accessing it. Access rights are specified additive and embrace read and write permissions. These ACLs for different schemas play an important role in our solution design; in particular, we discuss different alternatives for shared and private access rights during real-time collaboration in Section 6.4.

2.4.3 Exchange of Schemas

Similar to our solution design, FDBS exchange information about the schema. “Schema importation is the fundamental information-sharing operation in a federation. The term ‘importation’ refers to the process of [...] gaining access to some element of exported information.” [HM85]. Before exchanging information, each component is required to import the schema describing it. Schema exchanges are explicitly negotiated. Each component in a FDBS can discover other components via the ‘*federal dictionary*’. This federal dictionary provides names and network addresses of the components [HM85]. Thereafter, the schema exchange takes place between the components. In the design presented in Chapter 5, we expect metamodel changes in different information sources.

2.4.4 Negotiation and Transactions

Each component within a FDBS has a negotiation database, where it maintains state information initiating, accessing, and finalizing a transaction in the federation [HM85]. Similar to this idea, we incorporate state information in a—by far less technical—transaction (cf. Section 5.1.2.5 on p. 137). The authors discuss system-level issues: concurrency control, nested transactions, and object passing. As these topics are beyond the scope of the present thesis, we refer to the original literature [HM85] and to [Da04, p. 76ff, p. 465ff] as well as [BHG87, chs. 3, 5].

In our solution design, this negotiation between technical systems can be mapped to human tasks. Thereby, we utilize tasks carrying identifiers, such as the network address as part of a Uniform Resource Identifier (URI), to identify and refer to information stored in external information systems.

2.4.5 Replication of Information and Provenance

In the federated architecture of Heimbigner and McLeod, objects always reside in the component in which they are created, but references to them may be passed to other components so that the objects can be manipulated remotely through a set of exported operators. Figure 2.10 gives an illustration of the example given by Heimbigner and McLeod a message between components. In our solution design, we do not export such operators; however, to a certain extent, we assume that humans execute operations on information. We equip tasks with the necessary information to carry out a change. Also a message has a far more technical purpose, both concepts have the same purpose, i.e. to manipulate information.

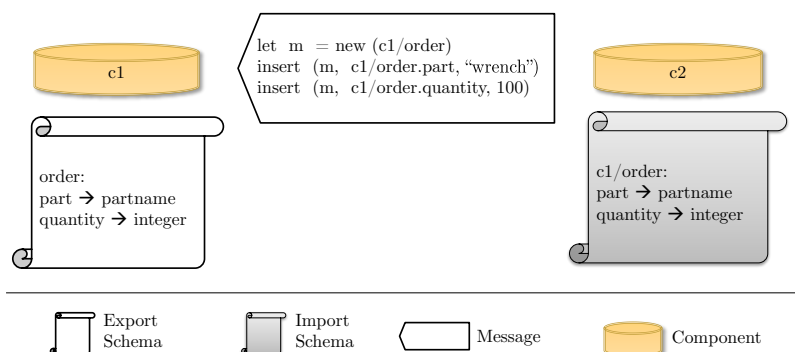


Figure 2.10: Message exchange between two components in a federated architecture

A copy of any shared (exported) object can be created by any component, but the copy is a different object, i.e. a transaction object. These transaction objects are equipped with unique names indicating the component that contains the specified object (cf. also ‘*surrogate key*’ in [KC04, p. 213ff]). In the remainder of the thesis, we utilize such a concept referring to it as *surrogate key* (cf. Section 5.1.2.1). The authors further outline how to update records in a federated architecture addressing the view update problem [BS81, CGT75, DB78, DH84]. We refer the interested reader to [HM85] for further details. In our solution design, we use tasks to update records in external systems, such that from a technical perspective a bidirectional integration is not required.

2.4.6 Integrity

In [PC05] and [Po07, ch. 5], global integrity constraints for FDBS are proposed. Although the authors claim that local components remain autonomous with their ‘*Active Component System*’ [PC05], they employ direct connections between DBMSs that participate within the federation. For EA model maintenance, this would require a considerable amount of modifications of existing systems. While we seek to reach an eventual consistent state within a federation (cf. Section 4.1.4), for our design we assume that modeling communities remain autonomously with respect to ownership, local responsibilities, etc. Moreover, our design omits modifications on information sources.

2.4.7 Evolution Process of Federated Database Systems

Introducing a FDBS is a complex task. Sheth and Larson [SL90] generalize the concepts of Heimbigner and McLeod and propose a reference architecture for FDBSs. This reference architecture can be utilized to compare prominent approaches to FDBSs. Moreover, they sketch the alternatives how to introduce a FDBS and provide a process that guides the introduction of a FDBS in organizations.

“The federation approach offers a preferred evolutionary path. It allows continued operation of existing applications to remain unchanged, preserves most of the

organizational structure, supports controlled integration of existing databases, and facilitates incorporation of new applications and new databases.” [SL90]

The authors highlight that a FDBS evolves through ‘*gradual integration*’ and divide the respective process in three phases:

- **Pre-integration:** This phase describes efforts that take place in order to migrate existing files to a DBMS in order to ease access. This includes
 1. the development of a local schema and component schema,
 2. loading the files in the DBMS, and
 3. modifying existing applications to access the DBMS instead of files.
- **Developing a FDBS:** The different schemas of a FDBS are created including component, export, federated, and external schemas. Further, this phase involves the definition of mappings between various schemas.
- **FDBS operation:** In this phase, the FDBS is put in operation and serves to manage and manipulate the integrated DBMS systems via accessing a federal system.

In Section 5.2 we describe an iterative method that covers some of these concepts. Especially the development and alignment of schemas in the first two phases is a challenging task for EA management.

Sheth and Larson [SL90] detail on how to develop schemas for a FDBS and distinguish between

- **bottom-up**, reusing existing DBMS to compose a FDBS, and
- **top-down**, introducing new DBMS to compose a FDBS.

The top-down approach is used only if the information demanded by the FDBS cannot be covered by existing DBMSs. In our approach, we assume a bottom-up approach. Technology-stacks of new applications are commonly not aligned with an EA repository.

2.4.8 Summary of Federated Database Systems

Different approaches to federated databases of information systems have been revisited. These approaches feature mechanisms to share and exchange data. Primarily, this is accomplished by an automated negotiation [HM85]. Common to our approach is the idea to share parts of information sources that are still managed autonomously, i.e. the structure of the system of systems is similar. Similar to FDBSs, we intend to share information among information sources. In contrast to FDBSs, we do not intend to maintain a bidirectional relationship to the original data. Although the FDBS solutions are technical and do not involve user interventions, one can learn from many problems described by the FDBS community (see e.g. [Co97]).

Contrary to FDBSs, we rely on expert knowledge as well as organizational accountability and responsibility for information. We automate and facilitate the import and conflict resolution but do not propose mechanisms for a technical negotiation among information systems. Our solution primarily intends to foster collaboration in order to maintain a federated model and improve consistency within and across local models.

2.5 Summary

In this chapter, we gave an overview of the problem domain and sketched important concepts of solution domains to build a general understanding of concepts that are used in the remainder of the present thesis.

We drew a parallel between model merging and software merging. These research communities cope with similar problems that arise when integrating different models into a consistent model. Commonly only one person is in charge of the merge process. Recent work also focuses on collaborative aspects as well as it introduces the notion of a tentative merge result. Both, the model merging and software merging communities, focus on the integration of a formally consistent artifact. In the remainder of the thesis, we emphasize the importance of consistency. However, EA management and their stakeholders can employ EA models to create value although these may be flawed with inconsistencies. We conclude that a high degree of inconsistency in EA models is not desirable, but (formal) consistency is not as important for EA management as for the model merging and software merging communities.

Thereafter, we drew a parallel to FDBSs. The properties of this system of systems, e.g. autonomy of each component, are similar to what we experience in organizations that maintain an EA model. The FDBS community assumes that different heterogeneous information systems exist prior to their integration and provides concepts to overcome heterogeneity in order to create federal views. In contrast to EA management, which often fosters to reduce heterogeneity, the FDBS community does not address such issues.

STATE-OF-THE-ART IN EA MODEL MAINTENANCE

So far, we have established an understanding of models and metamodels. For EA management, gathered information builds the foundation for profound decisions backed by information stored within a coherent EA model. However, as of today, EA model maintenance endeavors are considered error-prone and time-intensive by many enterprises [RHF⁺13, HMR12, HSR⁺13, FBH⁺13]. In this section we revisit the state-of-the-art in EA model maintenance.

DEFINITION 3.1: EA model maintenance[†]

EA model maintenance describes the implicit or explicit proposition to gather information on an EA.

[†]Synonym(s): describe an EA [BDM⁺10], EA documentation [FBH⁺13] ■

To provide a comprehensive overview of the different fields, literature reviews are performed according to the guidelines for an extensive literature review provided by Webster and Watson [WW02]. The literature review comprises leading journals and conferences of both communities, information system and computer science. As proposed by Webster and Watson, our “literature review is concept-centric” [WW02]. Thus, found concepts from literature determine the framework for our review.

Our review on EA model maintenance concentrates on giving an overview on the state-of-the-art of the *solution domain*. Recent research, e.g. [FBH⁺13, HMR12, RHF⁺13], coined the term automated EA documentation. Technical details, relevant information sources, data quality aspects, documentation processes, and respective challenges have been investigated by different research groups, e.g. Farwick et al. [FAB⁺11a], Buschle et al. [BHS⁺12], or Roth et al. [RHM13b]. Further, requirements [FAB⁺11b], governance

and processes [FAW07, FAB⁺11a], case studies [FBH⁺13], and issues [HMR12] are pointed out by different research groups.

Before we proceed with the presentation of the results of a rigorous literature study, we summarize key findings of a recent analysis of related work performed by Farwick et al. [FSB⁺14] and add additional insights we found during our literature study. Thereby, we employ a topic map to provide a mapping of Farwick’s study to our findings and to point out research gaps identified as well as contributions made by the present thesis and the published work that lead to it.

3.1 Research Topics in EA Model Maintenance

Figure 3.1 depicts a topic map that summarizes important aspects of EA model maintenance. We do not claim to solve all identified research gaps; however, we provide insights and contributions which shed light on these issues.

In the following, we outline the findings of Farwick et al. [FSB⁺14] denoted 1–8 in our topic map depicted in Figure 3.1. Thereby, we stick with the terminology as used by Farwick et al. [FSB⁺14], comment on some aspects and add further sources. Subsequently, we discuss each top-level topic and provide further details.

- 1. Interviews & forms:** Farwick et al. [FSB⁺14] describe this type as the most common way to collect information to maintain an EA model. Typically, information is gathered during an interview with stakeholders. Such interviews can be used to hand forms to stakeholders such that they can provide information in a semi-structured manner. Farwick et al. argue that only practitioners suggest such an approach [EHH⁺08, ASM⁺12, La13, Ke12, Ha10]. In [RHF⁺13], we outline that this ‘traditional’ approach to gather information for an EA model requires a high degree of manual work and, thus, is regarded error-prone, time-consuming, and cost-intensive. This goes in line with Farwick et al.; they note that a sole form-based or interview-based approach is rarely sufficient. Against this background, many publications seek to improve the situation.
- 2. Wiki collaboration:** This category describes methods that leverage Web 2.0 technologies to get information from stakeholders [FSB⁺14]. Many authors propose such an approach [BMN⁺11, FKF08, HS08]. The core idea is to keep the repository up-to-date by including many stakeholders. These stakeholders on the other hand provide information in an informal manner. Results from informal modeling can be understood by humans and—if necessary—can be further processed and formalized (cf. [Ne12, p. 42ff]). Even for traditional software engineering, Bruegge [Br13] argues that informal modeling is ‘OK’. Fiedler et al. [FHS⁺13] propose the integration of Enterprise Wikis into an EA repository and provide empirical grounds. Farwick et al. distinguish between approaches that employ semantic wikis such as [FKF08, HS08] and hybrid wikis that store structured and unstructured data [BMN⁺11, Ne12].

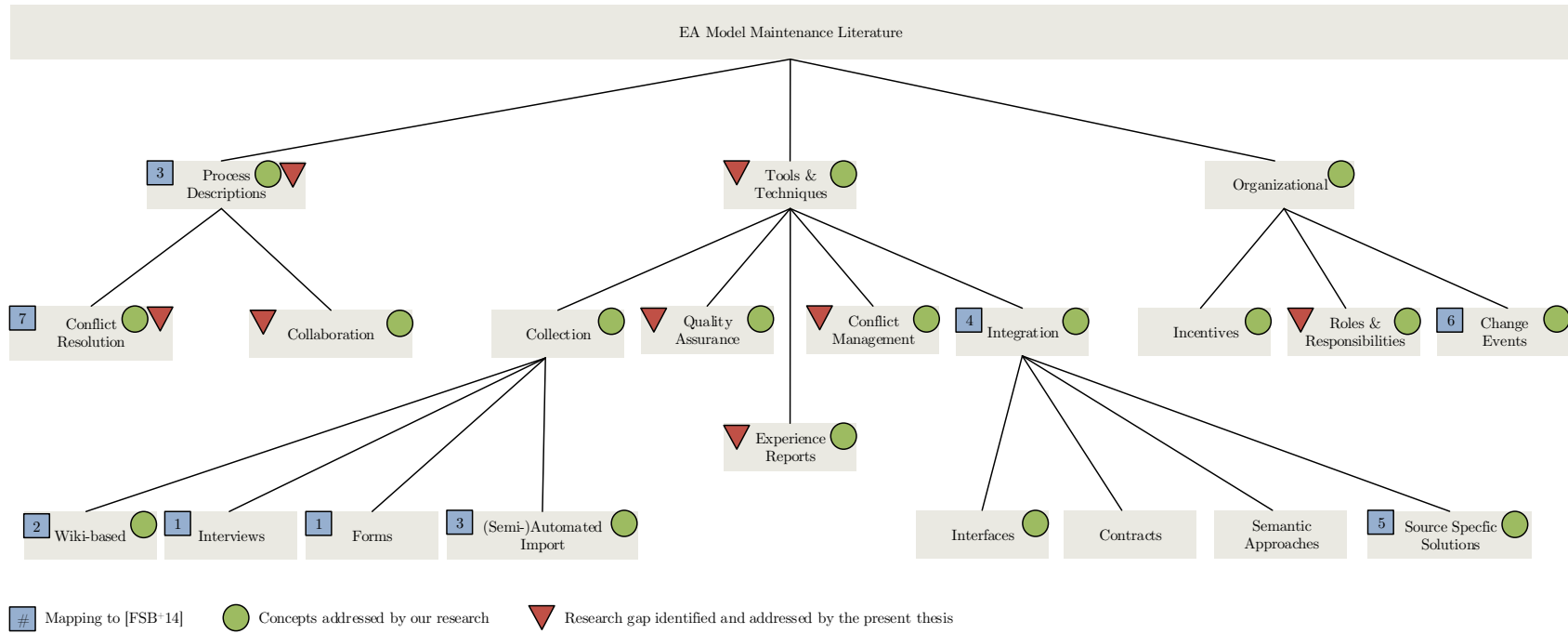


Figure 3.1: A topic map of EA model maintenance research

- 3. Defined data collection processes:** In [FAB⁺11a], Farwick et al. argue that the information to maintain an EA model is gathered in an ad hoc manner. In their more recent article, Farwick et al. [FSB⁺14] reports on work that prescribes pre-defined processes. Typically these processes come with a role description that (cf. Section 3.2.2 and Section 4.1.1) to indicate who carries out which activity [MJB⁺09, FAW07, FSB⁺12, FPB⁺12, FAB⁺11a]. Farwick et al. [FSB⁺14] close the discussion noting that none of these approaches discusses the organization-specific adaption of the processes. In the remainder of the paper [FSB⁺14], they introduce such a situational method. The idea is based on method engineering by Brinkkemper [Br96]. It can be roughly described with the core activities 1) characterizing the situation or project, 2) selection of method fragments from a method base, 3) assembly of method fragments (cf. also [BSH98]), and 4) measuring the project’s performance such that an administrative method align the method base. The last step serves as an iterative step to contribute and improve to the method base over time.
- 4. Generic import concepts:** In Section 1.1, we argue that many information that is subject of interest for enterprise architects is already existing within an enterprise. Further, this information is already contained in information systems that store information in formal models. Following this idea, researchers as well as practitioners propose to import information to the EA repository [SGT⁺11, PLW⁺12, Ke12, Ha10, FKF08, FHK09, FAB⁺11a, BGP11]. In their article, Farwick et al. note that some papers provide ideas how components of such a tool should look like [FKF08, PLW⁺12]. They further state that the “majority, however, stays vague on how to tackle technical challenges such as data mapping or the avoidance of duplicates.” [FSB⁺14].
- 5. Tool-, model, semantic integration:** Surprisingly, in the article by Farwick et al. [FSB⁺14] importing information is discussed prior to integration issues. As we point out in Section 4.2.2 and Section 5.2.2, this is a major challenge when reusing existing information of communities. Farwick et al. summarize approaches that seek to integrate existing tools or models with the EA repository [SOB⁺08, DL04, La13, CHL⁺13, ABB⁺07]. They provide a list of approaches that “focus on the integration of the content of diverse modeling tools into the main repository fostering a seamless navigation between the tools” [FSB⁺14], i.e. ter Doest [[DL04], Arbab et al. [ABB⁺07] and Lankhorst [La13]. Semantic technologies [BLHL01] are employed by Chen et al. [CHL⁺13] as well as Schmidt et al. [SOB⁺08] for the model integration. For a comparison on both alternatives, we refer to Table 4.6 on p. 100.
- 6. Automation via specific data sources:** When carrying out model-to-model transformations, one must understand the source and target models. Some work has been published on concrete models that have been used to extract information and map it to an EA repository. Examples are [HBL⁺12, BEG⁺12, AV10]. Alegria et al. [AV10], Buschle et al. [BHS⁺12], and Holm et al. [HBL⁺12] utilize network analysis tools to gather IT-infrastructure information. In collaboration with KTH Stockholm, we took a different approach. In Buschle et al. [BEG⁺12], we extracted information from an Enterprise Service Bus (ESB) to get information flows between business applications¹. Farwick et al. found that common to these approaches is there

¹For an extensive documentation we also refer to Grunow [Gr12].

limitation in scope, i.e. they are limited to a specific layer of the EA model. Further, they do not include methods on how to ensure the quality of the collected data. In addition to their conclusions, we note that these approaches assumed a specific target model, i.e. EA metamodel, which commonly is regarded to be configured in an organization-specific manner. Although we did not sketch how to improve the situation, in Grunow et al. [GMR12], we published empirical insights of data quality aspects of ESB systems.

- 7. Change events and notifications:** Farwick et al. continue their review of related work with publications that discuss the notion of triggering events for EA model maintenance. These events are meant to indicate relevant changes [ASM⁺12, FSB⁺12, SGT⁺11, Ha10, Bu11]. Farwick et al. provide examples of publications that only mention the importance of external events [ASM⁺12, SGT⁺11, Ha10]. In her PhD thesis, Buckl considers triggers in the provided metamodel [Bu11, p. 67]. Farwick et al. close the discussion by referring to their paper [FPB⁺12] that discusses implementation issues related to the utilization of external events to trigger EA model maintenance processes.
- 8. Conflict resolution & quality assurance:** The final category found by Farwick et al. [FSB⁺14] considers the resolution of conflicts that arise when integrating different information sources. Thereby, authors consider quality assurance steps and automation [FAW07, RHM13b, FAB⁺11a, FPB⁺12, MJB⁺09]. Farwick et al. suppose that only one of these papers reveals details and refer to their paper [FPB⁺12] that presents implementation details for conflict resolution mechanisms. Especially in [RHM13b], we propose a concrete process and visual means that has been demonstrated in [HRP⁺13b]. We proposed processes, role description, design, and a prototypical implementation on the evolution and resolution of conflicts that build the foundation for the present thesis [RHM13b, RHM13a].

Farwick et al. summarize their findings during the literature study and state the existence of ‘*small islands*’ of research on EA model maintenance which are carried out by few different researchers and conclude that none of these combines EA model maintenance with situational method engineering. As stated above, Farwick et al. then introduce such a situational approach in their paper. Their situational method is based on the ideas of Brinkkemper [Br96] and allows to characterize the situation and assemble a method that is deemed appropriate for an organization. Farwick et al. provide software-support for their method and use an Resource Description Framework (RDF) store to persist information. They present insights of a case study and confirm that a software-supported process can provide utility to organizations.

The approach of Farwick et al. comes with two major drawbacks: First, they do not address updates in an EA repository that may be in conflict with changes within the information sources of other modeling communities. We claim that conflicts arise in the course of concurrent modeling activities within multiple information sources describing the same real-world object. This includes modeling activities within the EA repository and respective information sources. Second, Farwick et al. assume that an EA metamodel does not change over time. In [RHM13a], we stress that a coupled model and metamodel evolution is an important property for a successful EA initiative. It allows to start lean, to show results

of EA management early on, and to extend the EA repository incrementally. Especially, extending the metamodel of the repository allows to address new concerns of stakeholders (cf. Section 2.1.2 and Section 2.1.4). In the remainder of this thesis, we present a design that addresses these issues thoroughly.

The following sections are dedicated to provide further insights in the state-of-the-art of EA model maintenance and important details that influenced our work. Thereby, we employ the structure of our topic map illustrated in Figure 3.1.

3.2 Organizational Aspects

Many authors write on organizational aspects of EA and EA management, e.g. Ahlemann [ASM⁺12, ch.9], Hanschke [Ha10, ch.6]. They highlight the importance of stakeholder analyses, role definitions and responsibilities. In his master’s thesis, Aleatrati [AK14] investigates specific organizational aspects that are especially important to maintain an EA model. In Figure 3.1, we illustrate these aspects: incentives, roles, and change events.

3.2.1 Incentives

Incentives play an important role for EA model maintenance and EA management in general. In line with Ahlemann et al. [ASM⁺12, p.49], Aleatrati [AK14] confirms that incentives and supportive stakeholders are of utmost importance for a successful EA model maintenance and, hence, for a successful EA management. Thereby, the authors describe two fundamentally different ways (cf. also Ross et al. [RWR06, pp.132–134]):

- **Top down via organizational measures:** This strategy commonly incorporates the EA model maintenance tasks in objectives of individuals. Ahlemann et al. [ASM⁺12, p.245] state this common management practice as a means to accelerate the introduction of EA management in an organization. During our interview series Section 7.5 we could confirm the work of Ahlemann et al.: A common practice to increase support of individuals couples goals to financial compensation. In particular, we found instances in which EA model maintenance success has been coupled with bonuses. Additionally, Ahlemann et al. [ASM⁺12, p.245] list other forms of compensations such as:
 - attending an EA management training,
 - visiting courses for EA management (or related) certifications, or
 - participating at EA management conferences.
- **Via prestige:** Ahlemann et al. recommend to rewards ground-breaking EA-related projects with ‘*EAM awards*’ [ASM⁺12, p.245]. Delivered during a social event such as Christmas or annual meetings such an award is not limited to its motivational effects but also serves to inform about the EA initiative and can “be exploited for marketing purposes” [ASM⁺12, p.245]. In the interview series of Aleatrati [AK14],

we could not confirm this approach. However, we regard this a meaningful way to promote an EA initiative.

Besides these incentives, we propose a bottom-up approach to EA management (cf. Section 2.1.4) that is able to show results early. In [RHM13a], we propose an evolutionary approach to EA modeling that seeks to increase stakeholder buy-in. Our approach falls in the ‘*wiki collaboration*’ category of Farwick et al. A general trend we diagnose in [RM14] is that current EA repository seek to lower technical barriers to access information, e.g. via web-based platforms to increase stakeholder buy-in. In our approach [RHM13a], we employ the notion of tasks to engage stakeholders in the evolution of an EA model. However, empirical research which provides evidence confirming that lowering the technical barrier to contribute to an EA model actually has an impact on stakeholder buy-in is yet to be done.

3.2.2 Roles

Besides a process model, Fischer et al. [FAW07] introduce role descriptions and a mapping of process activities to roles and their responsibilities. Thereto, Fischer et al. uses a Responsible Accountable Consulted Informed (RACI) matrix. Other work builds upon these role descriptions, e.g. [FAB⁺11a, FSB⁺14, RHM13a]. We sketch the roles introduced by Fischer et al. briefly and refer to Section 4.1.1 for a more comprehensive role description:

Chief enterprise architect is not directly involved and only informed about updates on the EA model. The management is carried out by the EA coordinator.

EA coordinator manages the EA model maintenance. The EA Coordinator

- enacts the EA metamodel,
- supports the specification of interfaces to specialized repositories and their models,
- maintenance of information stored within the EA repository, and
- is responsible for the compilation and design of reports on the EA.

Data owner provides information to the EA team. Each specialized model (modeling community), has a data owner. Data owners assist the EA team in specifying and maintaining the interface between the EA repository and the specialized model.

EA stakeholder subsumes all personnel within business and IT units that contribute or utilize EA information.

EA repository manager is a technology-oriented role (cf. also [RHM13a]) responsible for technical issues related to the EA repository.

3.2.3 Change Events

The final organizational aspect centers around ‘*change events*’ or ‘*triggering events*’. As outlined above, events in the real-world commonly trigger EA model maintenance endeavors. For the maintenance of an EA model, Fischer et al. [FAW07] discuss two different strategies to initiate a new maintenance cycle, these are:

periodic is initiated by the EA team and based on a maintenance schedule. Fischer et al. use their contract based approach to incorporate the schedule in this contract. In their periodic approach, the EA team triggers a data owner. The data owner then provides the model of an information source as defined in the contract.

non-periodic can be triggered by both parties, the EA team as well as data owners. A non-periodic cycle is initiated, e.g. if models changed significantly. Fischer et al. provide an example: The model could change due to project work. Upon project completion, the data owner informs the EA team about the changes which then decides whether or not to initiate a maintenance cycle for the respective information source.

We refer to Section 4.2.4 during which we discuss the different notions of these events in more detail. During the discussion, we draw a parallel to other disciplines like data warehouse (DWH) research.

Moreover, we refer to other authors who provide a concrete list of events. In [FSB⁺12], Farwick et al. present a non-exhaustive list of change events. To some extent, the authors assume that these events are triggered by tools rather than persons. However, they also provide the real-world event, e.g. ‘*project start*’ is triggered by the PPM tool. In Assumption 4.3 on p. 66 we state that a bi-directional tool integration is beyond the scope of the present thesis. In Chapter 7 we summarize feedback from practitioners which confirms our hypothesis that it is too cost intensive for EA practitioners and comes with additional technical difficulties.

3.3 Process Descriptions

Jonkers et al. state that “instruments needed for creating and using enterprise architectures are still in their infancy” [JLD⁺06]. They further elaborate that EA frameworks provide high-level guidance in identifying which areas (cf. concern in Section 2.1.2) of business and technology should be considered when creating an EA and point out that current EA frameworks, e.g. TOGAF [Th11] and Zachman [Za87] provide little assistance in creating the architectural artifacts themselves. We diagnose that as of today, the situation has not changed; this is confirmed in a more recent paper by Farwick et al. [FSB⁺14]. Since then, some authors address the maintenance process in more detail seeking to improve the situation. We report on these efforts, outline their core contributions, and—if applicable—sketch how they have influenced our research.

Fischer et al. [FAW07] present a federated approach to keep EA models up-to-date. Thereby, they designate the EA repository “to store a copy of model data from specialized architectures

relevant for EA purposes”. Fischer et al. detail their concept to maintain an EA model; that includes a high-level maintenance process and role descriptions. Moreover, they discuss shortcomings of existing approaches to EA model maintenance and reports on the application of their approach at a financial service provider. We briefly summarize the key points of their approach, findings in literature as well as insights Fischer et al. gained during the application of their approach.

- **Reuse of existing models:** In line with ter Doest and Lankhorst [DL04], Fischer et al. agree “that EA modeling should focus on consolidating models, modeling techniques and tools already existing in a company and integrating these at an appropriate level of abstraction” [FAW07], i.e. they propose to reuse existing models of more specialized architectural descriptions for EA model maintenance. This has been adopted by other research communities, e.g. Buschle et al. [BEG⁺12], Farwick et al. [FAB⁺11a], and Moser et al. [MJB⁺09] and practitioners (cf. [RHF⁺13]). Thereby, specialized communities remain autonomously, i.e. are in full control, of their model.
- **Scope and evolution:** Fischer et al. [FAW07] propose to specify and maintain interfaces between EA and specialized architectures and provide a list of concrete information demands. In contrast to Fischer et al. [FAW07], we present an iterative approach that is guided by concrete concerns (cf. Section 2.1.2 and Section 2.1.4). This goes in line with the anti-patterns detailed by Ambler et al. (cf. ‘*modeling for the modeling’s sake*’ and ‘*yesterday’s enterprise model*’ in [ANV05, pp. 142–143]) and accounts for the typical characteristics of EA management endeavors we observed that create little value during long-lasting EA model maintenance endeavors [HMR12, HRS⁺14]. Although they provide a list of entities for information demands of an EA model, Fischer et al. conclude their paper with the insight that the integration of information from specialized architectures into an EA repository is an ongoing process rather than a one-time effort. They further state that it “is necessary to monitor the quality of model data from source systems continuously—particularly regarding their consistency” [FAW07].
- **Autonomy of models:** “In real life, several models for different parts of the enterprise might be maintained, and/or EA will co-exist with other, more specialized architectures that cover a subset of those artifacts” [FAW07]. This autonomy is one of our main assumptions that influenced our design (cf. Assumption 1.1 and Definition 1.1).
- **Conflicts and inconsistencies:** stakeholders are involved in the resolution of inconsistencies. Although Fischer et al. [FAW07] do not directly refer to these inconsistencies as conflicts between different models, we assume that these inconsistencies are model conflicts. “The integration of model data from specialized architectures into the EA repository is an ongoing process rather than a one-time effort. It is necessary to monitor the quality of model data from source systems continuously—particularly regarding their consistency” [FAW07].
- **Iterative stakeholder involvement:** Fischer et al. [FAW07] propose to involve stakeholders in the maintenance process. Fischer et al. propose that all changes intended to be performed on the EA repository are compiled to a report prior to their

application. All EA stakeholders are then informed about intended changes and are meant to evaluate them. Eventually, EA stakeholders may state their objection with an explicit veto. The EA team coordinates the vetoed stakeholder and the data owner and—if necessary—other affected stakeholders. This goes in line with Hanschke [Ha10, p. 101]; she also advocates that maintenance of an EA model should be performed by people who have the knowledge. Others follow this idea, e.g. [FAB⁺11a, RHM13b].

- **Final approval by a responsible person:** Fischer et al. introduced the notion of a final approval for changes on the EA repository. Since it serves as a decision base, it must contain reliable and consistent information. However, initial feedback from practitioners has been quite controversial (cf. [RHM13b]). While some practitioners prefer the final approval, there are others that refer to this step as an act of ‘unnecessary bureaucracy’ which is perceived as an ‘administrative barrier’ [RHM13b]. A situational approach (e.g. Farwick et al. [FSB⁺14]) might help. Initially, Farwick et al. [FAB⁺11a] distinguished between major and minor releases. The former must be approved by a person before it is applied to the EA repository whereas the latter immediately is applied to the EA repository.

Next to the details of the process descriptions provided by literature, we briefly discuss a specialized process description that copes with conflicts.

3.3.1 Conflict Resolution

Fischer et al. [FAW07] and Farwick et al. [FAB⁺11a] propose processes for EA model maintenance which incorporate high-level activities that are meant to resolve inconsistencies and conflicts. In [RHM13b], we detail this high-level process description (cf. also Section 5.2) and provide further insights. In line with Farwick et al. [FAB⁺11a], we distinguish between minor and major updates [RHM13b]. During the evaluation of the process [RHM13b], we show that it still remains unclear how such a process looks like. It strongly depends on the organizational context (cf. [FSB⁺14]) how exactly the conflict resolution is guided by a process. During our interview series (cf. Section 7.5), most organizations follow an ad hoc process. Thus, we conclude that respective software support for EA model maintenance must facilitate non-deterministic processes (cf. also REQUIREMENT PR1 on p. 119).

Inspired by Farwick et al. [FAB⁺11a], we employ tasks to resolve conflicts. In [RHM13b], we present details of the conflict resolution process. In the proposed process, the EA repository manager creates a new *conflict task* if the conflict cannot be resolved. This conflict task contains any information about the conflicting model elements and may provide additional tool support, e.g. a specialized merge tool, to facilitate the conflict resolution. The EA repository manager is able to understand the model conflict on a conceptual level but, however, is unable to provide required information on an instance level. Thus, in the next step, the data owner receives the conflict set including the respective configuration. This way, the process proposed in [RHM13b] contains an escalation mechanism to get the information. This characteristic has been incorporated in the present thesis. In contrast to the approach of Farwick et al. [FAB⁺11a], we facilitate task support by incorporating the tasks in interactive visualizations. In the initial evaluation, practitioners liked the idea

to provide interactive visual means to resolve a conflict. They referred to ‘*gamification scenarios*’ [RHM13a].

3.3.2 Collaboration

Initial thoughts on collaboration come from Fischer et al. [FAW07]. The authors propose to involve EA stakeholders and data owners in the EA model maintenance process. Farwick et al. follow this idea in [FAB⁺11a]. They detail how to integrate new information sources with an EA repository. In their process design, the EA repository manager and the data owners are supposed to collaboratively resolve arising conflicts. However, both research groups do not detail the conflict resolution process. In [RHM13b], we present a software-supported process design that describes this collaboration during the resolution of conflicts in detail. Initial concepts and practitioner feedback presented in [RHM13b] build the foundation for the present thesis.

3.4 Tools and Techniques

Many publications report that tool support is insufficient, e.g. [FAW07] In a more recent paper, we [HMR12] confirm this issue and report on arising challenges when automating EA model maintenance. These findings are incorporated in the use case analyses and requirements elicitation (cf. Section 4.2 and Section 4.3).

3.4.1 Collection

As discussed above and reported in the state-of-the-art section in their article, Farwick et al. distinguish between:

- Interviews and form-based and
- Wiki-based approaches.

In addition, we see publications of another category, namely *semi-automated imports*. This refers to the import of integrated models in an EA repository. That includes the import of models that are build automatically by network scanners [AV10, HBL⁺12, BHS⁺12, BBK⁺13]. Their approaches commonly focus on technical information about an EA, e.g. available servers and operating systems.

3.4.2 Experience Reports

Initial findings on experience reports are given by Fischer et al. [FAW07] and Farwick et al. [FSB⁺14]. Farwick et al. present their approach that builds on four documentation techniques which are meant to be configured in an organization-specific EA model maintenance process. Their approach is prototypically implemented and has been applied at

a German insurance company. The authors report on findings from this case study in particular. In his master’s thesis, Kirschner [Ki14] presents two case studies. He reports on the application of the design related to the present thesis and reveals arising challenges as well as solutions. The essential setup and most important findings are summarized in Section 7.2 and Section 7.3. Other empirical reports are based on surveys or interviews [FBH⁺13, RHF⁺13, RHM13b]. As of today, studies that investigate the approaches at a broader scale are missing.

3.4.3 Quality Assurance

Fischer et al. [FAW07] introduce the concept of ‘*data delivery contracts*’. Such a contract is intended to ensure the data quality delivered by a third party (cf. data owner in Section 4.1.1.2). “A data delivery contract includes a definition of the interface to the source system, descriptions of model data from the specialized architecture to be stored in the EA repository, transformation rules and a maintenance schedule. Data maintenance processes are executed in regular intervals. Special events however, may trigger additional maintenance cycles. Before model data from specialized architectures are stored in the EA repository, consistency checks are performed” [FAW07]. This research gap is also emphasized by Farwick et al. in [FSB⁺14]. In [RM14], we introduce a concept to show visual model and metamodel differences (cf. also Section 5.3.1).

3.4.4 Conflict Management

We diagnose that EA literature is scarce on tool support or designs for the collaborative resolution of model conflicts. While many authors state that it is important to cope with conflicts and call for automation and tool support, e.g. Farwick et al. [FAB⁺11a] or Fischer et al. [FAW07], only little has been published that sketch how conflict resolution could be supported by software, e.g. [RHM13b, HRP⁺13b].

3.4.5 Integration

A first step towards gathering information from existing models is integration (cf. Section 4.2.2 and Section 5.2). With respect to EA model maintenance, literature distinguishes between integration via:

- **Interfaces:** Some authors, e.g. [BEG⁺12, HMR12], propose interfaces to other models. This requires a physical integration; whether loosely coupled or not. In the context of EA management, physical integration approaches are discussed by Engels et al. [EHH⁺08, p. 206ff]. The authors distinguish between
 - presentation
 - logic, and
 - data integration.

- **Contracts:** Fischer et al. [FAW07] suggest a contract that specifies a model that is then transferred as a snapshot, e.g. in a Comma-Separated Values (CSV) file.
- **Semantic matching:** A more sophisticated variant for integrating different models with each other is proposed by Chen et al. [CHL⁺13]. The authors propose to employ semantic technologies [BLHL01], i.e. ontologies, to map different models.
- **Source specific models:** Some approaches provide insights on specific models, e.g. [BEG⁺12]. In [BEG⁺12], the outcomes from a practical application of a concrete mapping between a model of an Enterprise Service Bus (ESB) available as COTS product and an EA model that conforms to the ArchiMate 2.0 [Th12a] metamodel is presented. While there is little to learn for our research, these kind of contributions are interesting for practitioners and show feasibility as well as they reveal technical challenges.

Although further approaches to integration of information systems exist (see [Co97, p. 84]), to our best knowledge only above outlined methods and techniques have been applied to automate EA model maintenance. In Section 5.2.2, we describe our notion of integration based on dimensions proposed by Frank [Fr08] and Kattenstroth et al. [KFH13].

3.5 Summary

We reviewed literature on EA model maintenance and gave an overview of insights that influenced our design. Although current EA frameworks, e.g. [Th11, De09, Za87], mention that EA model maintenance is important for a successful EA management initiative, these frameworks do not provide any details on how to procure information efficiently. In line with the recently published journal article by Farwick et al. [FSB⁺14], we conclude that only few research communities investigated the topic and a coherent design for an integration of different models into a holistic EA model is missing.

We discussed issues arising when integrating different federated models with an EA repository and presented multiple empirical studies on the topic, e.g. [HMR12, FBH⁺13, RHF⁺13]. A topic map for EA model maintenance served to sketch the research gaps. While we do not claim to solve all of these issues, some of them are addressed by the work presented in this thesis.

In the present thesis, we detail the conceptual foundations including typical characteristics of EA model maintenance endeavors which seek to integrate multiple federated models with a single EA model. We propose a design for collaborative model integration that includes conflict resolution and quality assurance. This includes a process design, role descriptions as well as software support. Our design combines multiple fields related to the topic in a coherent manner (cf. Section 2.3, Section 2.4, and Chapter 5). Before we detail our design, we describe the core use cases and requirements in the next chapter.

REQUIREMENTS ANALYSIS

In this chapter, we detail our initial observations leading towards our design decisions. For a better understanding, we sketch the essentials of the solution space for Federated EA Model Management subsequently. These are supplemented by the design details which we describe in the next chapter. In this chapter, we further present use cases and requirements for MODELGLUE—the software support for Federated EA Model Management. To present the use cases of various stakeholders in Federated EA Model Management, we employ a structured use case template. The structure of this template is explained followed by the results of a rigorous use case analysis. Afterwards, we derive concrete requirements utilizing the identified use cases as a foundation. In the final part of this chapter, we summarize use cases and requirements and relate them to each other.

4.1 Conceptual Overview of Federated EA Model Management

In this section, we present a high-level overview of our conceptual design to provide the reader with knowledge to foster an understanding of use cases relevant to federated EA model environments and Federated EA Model Management.

Figure 4.1 illustrates a conceptual overview of a federated EA model environment. Modeling communities, e.g. PPM, BPM, and ITSM, make up a federation with the EA management community as the federal entity (cf. **A**, **B**, **C**, and **D** in Figure 4.1). Each community performs tasks which follow processes defined either explicitly or implicitly. These processes are supported by technology. In [BMR⁺10a] we report that each modeling community can be considered a separate linguistic community. That means each community describes real-world objects with their own terminology. Although the modeling communities may refer to the same real-world objects, they use different names and attributes to describe them.

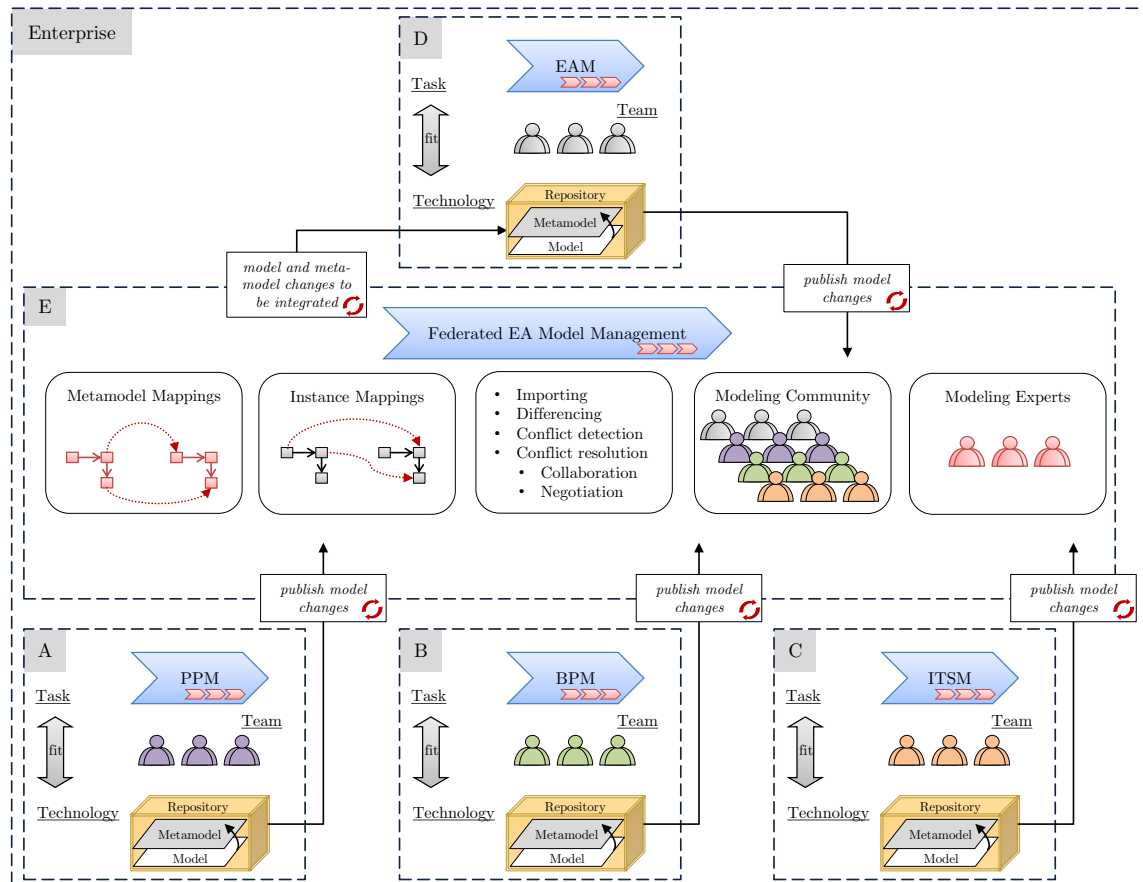


Figure 4.1: Conceptual overview of Federated EA Model Management as a socio-technical system of systems

This is also reflected by respective conceptualizations, i.e. the metamodels of the modeling communities exhibit different structures and use various terminologies. As motivated in Section 1.1, multiple *modeling communities* perform their tasks employing highly specialized repositories which capture information in a model. For the further discourse we establish a special notion of consistency of repositories within a federation in Definition 4.1.

DEFINITION 4.1: Local consistency

The repository of a modeling community in a federated EA model environment is *locally consistent* if information captured in the model conforms to a metamodel, i.e. the repository's model is consistent with its metamodel. ■

Commonly, best-practice knowledge is incorporated in these repositories. This influences the design of their metamodels; i.e. the specificities of a modeling community are incorporated in a metamodel and even may change the way software is developed, cf. Domain-Specific Modeling (DSM) in [KT08, p. 15]. In [GT95], Goodhue and Thompson observe a phenomenon and coined the phrase '*task technology fit*'. The authors highlight that individual performance rises if a good fit prevails between employed technologies and the task. Hence,

we conclude that the repositories employed by the different communities are intended to fit best for the tasks at hand for this particular community.

A different terminology, best-practice processes, efficient communication within a community, and convenience are major reasons why modeling communities want to stay autonomously. Hence, we articulate Assumption 4.1 for our approach.

ASSUMPTION 4.1: Modeling communities

We assume the modeling communities do not adhere to a top-down terminology and metamodel(s). Further, we assume that each modeling community wants to remain autonomously.

Our next observation is that IT environments of organizations are large systems of systems that are commonly not build from scratch [Za97]. In particular approaches to EA management commonly do not assume to build a system of systems from scratch, i.e. enterprise architects do not build an EA with a greenfield approach. Especially in a large scale enterprise (LSE), a plethora of business applications exists; a greenfield approach, hence, is rather unusual for today's organizations. The—commonly heterogeneous—technological base of an entire IT environment cannot be simply ripped and replaced with new technology. For the design of Federated EA Model Management, we presume characteristics of these repositories as explicated in Assumption 4.2.

ASSUMPTION 4.2: Repositories of modeling communities

We assume that a repository of a modeling community

- varies between a local consistent and inconsistent state,
- features a metamodel which is either stored within the repository explicitly or the implementation of the repository reflects the metamodel implicitly,
- may be based on legacy technology, and
- cannot be replaced with new technology due to technical or financial constraints.

The *teams* of modeling communities are aware that the information they maintain may contribute to the overall performance of an organization. As expressed more general in Assumption 1.1 (see p. 5), we presume that members of a modeling community are willing to publish and share information at a regular basis. This way, the community members can contribute to the decision base of EA management. Put differently, they contribute to the common good of an organization by sharing information with the federation (cf. **E** in Figure 4.1).

Since each modeling community is a separate linguistic community, their conceptualization has to be aligned. A conceptual alignment ensures that everyone knows and understands a unique concept by a certain term in a particular modeling community. The result of such

an alignment is a semantic mapping of different metamodels. To develop such a semantic mapping *modeling experts* have to align the understanding of different models. This requires a close collaboration among community teams and the EA team. The final outcome of these conceptual alignment efforts manifests in explicit *metamodel mappings*. These explicit metamodel mappings can be transformed in syntactic mappings that are executable by an information system.

Once an alignment of different concepts between the metamodel of a modeling community and the EA metamodel is agreed upon and the mapping is created, instances can be shared with the federation. This is commonly done by *importing* model changes to a *federal system* that serves as the glue between repositories of modeling communities and the EA repository as the technology support of the federal entity. This federal entity, i.e. holistic EA management function (cf. **D** in Figure 4.1), can utilize exchanged information within the federal system to optimize the organization as a whole, e.g. increase flexibility or reduce costs (cf. Section 2.1). Prior to that, a synchronization of model and metamodel changes of both information sources and EA repository must take place. The EA repository thereby serves as a sink for the information sources (cf. Assumption 4.3). Confirming our assumption, EA experts (cf. Chapter 7) perceive a bidirectional synchronization as too cost-intensive, atypical for a federated EA model environment, and fragile.

ASSUMPTION 4.3: Synchronization type

We assume a unidirectional synchronization between information sources and EA repository, i.e. information is exchanged from multiple information sources to the EA repository (information sink).

From a technological perspective, modeling communities can either *publish model snapshots* or *publish model changes*. In order to be able to store historical information of different integrated models, especially the former case requires calculating the differences between models. This process is called differencing. Differencing is very important to enterprise architects in different contexts (cf. Chapter 7). They want to compare differences between models within the EA repository and the federal system. The federal system stores the community models in a uniform manner, i.e. these models are—to a certain degree—consistent with a common metamodel.

The information exchange between modeling communities and the federation takes place at a regular basis as a unidirectional synchronization. Once integrated, information can be changed in its origin (the information source), but can also be modified by enterprise architects in an EA model. That is, modifications on information that describe the same real-world objects or properties thereof can take place in different models concurrently. In [KR14, RM14], we report that in the course of concurrent modeling activities conflicts may arise on three different levels: model/model, model/metamodel, and metamodel/metamodel conflicts.

In Example 4.1 we describe a situation that ends with a model/model conflict¹.

¹For now, we assume an intuitive understanding of the terms `ATTRIBUTE` and `ATTRIBUTEDEFINITION`. We refer to Section 5.1.2 for a description of these concepts.

EXAMPLE 4.1: Model/model conflict

The enterprise architect calls the infrastructure team. They respond that they do not host Linux servers anymore. Hence, the enterprise architect changes the ATTRIBUTE ‘OPERATING SYSTEM’ of the BUSINESS APPLICATION ‘SAP CRM’ to the default, i.e. ‘Windows Server 2012 R2’. At the same time, an infrastructure monitor detects the actual running instance of this ‘SAP CRM’ APPLICATION and updates information within the CMDB. The CMDB serves as an information source and is part of a federated EA model environment. Both, the EA model and the CMDB, describe the same real-world object but have concurrently maintained contradicting information about the operating system. Hence, a model/model conflict arises on the next synchronization of the CMDB (as the information source) with the EA repository.

The second kind of conflicts may occur between the model and its metamodel (cf. Example 4.2).

EXAMPLE 4.2: Model/metamodel conflict

A type ‘BUSINESS APPLICATION’ has an ATTRIBUTE ‘MAINTENANCE CYCLE’ indicating the frequency updates are installed. The system is optimized based on the current performance characteristics. An instance of such a BUSINESS APPLICATION is ‘SAP ERP Central Component (ECC) 6.0 Enhancement Package 7’ denoted SAP ERP in the following.

An EA modeling expert changes the ATTRIBUTEDEFINITION of ‘MAINTENANCE CYCLE’ from natural text to an enumeration {weekly, biweekly, monthly, quarterly, bi-annually, annually}. The EA modeling expert thoroughly considered implications of this single update operation of an ATTRIBUTEDEFINITION and applies it after carefully considering all literal descriptions and conversion of all instances. However, at the same time, a data owner changes the ATTRIBUTE ‘MAINTENANCE CYCLE’ of the SAP ERP from ‘two times a month’ to ‘once every month’. This concurrent change results in a conflict of the model (SAP ERP) and the metamodel (BUSINESS APPLICATION).

Typically, model/metamodel conflicts are resolved by altering the model. The third category of conflicts commonly cannot be resolved that straight forward and needs more collaboration, discussions, and synthesis. Example 4.3 illustrates such a situation.

EXAMPLE 4.3: metamodel/metamodel conflict

Let us assume the same situation as described in Example 4.2. Meanwhile, an enterprise architect deletes the `ATTRIBUTEDEFINITION` ‘MAINTENANCE CYCLE’. Then, a metamodel/metamodel conflict occurs between the changes of the EA modeling expert and the enterprise architect: While the EA modeling expert is concerned about a more thorough definition of the attribute, the enterprise architect wants to discard it—both changes cannot be applied to the same metamodel.

We conclude that in the course of such concurrent modifications *conflict detection* is necessary to identify conflicts that occur and to keep the models within a federation consistent. When conflicts are detected, it is necessary to resolve them to restore a consistent state within the federation. Typical characteristics of this *conflict resolution* process are that it is highly collaborative and long-lasting [RHM13b] compared to other approaches. For instance, in a distributed version control system (DVCS), the developer is the sole person obliged to resolve conflicts [Ch08, p. 28ff] and to submit a new version to the DVCS.

Although the technical and semi-automated synchronization between information source and EA repository takes place in a unidirectional manner, the members of a federated EA model environment want to establish and sustain a consistent state. To be able to update the imported instances in their origin, a mapping of the EA model to the initial information source is required. These mappings are called *instance mappings*. To distinguish one described object from another, in line with Khoshafian and Copeland [KC86a, KC86b] and Evans [Ev04], we formulate Definition 4.2.

DEFINITION 4.2: Identity

Identity of an object distinguishes it from all others [KC86a]. It is identity that helps to track an object through different states, across different implementations, and the real-world [Ev04, ch. 5]. ■

Summarizing our present understanding, autonomous modeling communities act as a part of a federated EA model environment and are willing to share, i.e. publish, information that is integrated—possibly in a transformed manner—with an EA model. Further conflict detection and identity reconciliation is necessary to inform the teams of the modeling communities about conflicts. These modeling communities are considered an essential part of the federation. In the following, we take a closer look at the communities and investigate essential roles within the teams.

4.1.1 Roles within Federated EA Model Management

Next, we characterize the members of a federated EA model environment such that the reader gains a better understanding of the involved stakeholders and identified use cases. In

this section, we use the term *role* in its sense as an organizational role (see Definition 4.3) not to be confused with the role concept of UML or modeling theory.

DEFINITION 4.3: Role[†]

A role is a comprehensive description of duties, responsibilities, and tasks one or more individuals should perform within an organization. Such an individual may be obligated to possess several roles. The assignment of roles to individuals is commonly based on their qualifications.

[†]Synonym(s): functional role [Be12, p. 24]; actor [OMG11b, p. 598] ■

Before we detail different roles within a federated EA model environment, our intent is to point out the general nature of these roles as well. We perceive the majority of the community members as *knowledge workers*. The main capital of a *knowledge worker*, e.g. software engineers and scientists, is knowledge, because they “think for a living” [Da05, ch. 1]. In contrast to other forms of work, the primary task of knowledge work is ‘non-routine’ problem solving involving a combination of convergent, divergent, and creative thinking [RSS⁺11]. This commonly requires a high level of education and the use of IT as an integral part of the work [Py05]. Grounds for an analogy to individuals within a team of a modeling community is given by Davenport [Da05, ch. 1], who highlights that knowledge workers prefer autonomy. In a federated EA model environment this not only holds true for each team as a whole, but also for the individual members. We conclude that the individuals within a federated EA model environment can be considered ‘*knowledge workers*’.

In contrast to the need for autonomy of each modeling community and respective individuals, there is also a demand for coherency [DGS⁺09, p. 494ff]. In EA management and Federated EA Model Management, the EA team is responsible to establish and sustain coherency between different models and the EA model. To accomplish this challenging task, management support is essential as detailed in Assumption 4.4 but also a major issue for most EA management endeavors [HSR⁺13].

ASSUMPTION 4.4: EA management function

We assume that EA management has been established within an organization and is valued and supported by upper management. This includes a managerial authority for members of the EA team, e.g. to issue compulsive directives to contribute information or correct model inconsistencies.

Besides upper management support [DGS⁺09, p. 499ff], stakeholder buy-in is of utmost importance for an EA management initiative in order to succeed [Ha10, p. 97ff]. Thereby, the identification of stakeholders is crucial. Austin et al. [ANO09, p. 203ff] presents a three-step framework to identify stakeholders assessing their importance and influence as well as determining their interests and motivations. Depending on the outcome of the analysis, Austin et al. provide guidelines how to deal with these stakeholders. An even more comprehensive guide is given by Belbin [Be12, p. 22]. He describes different general

‘*team roles*’ from a social perspective. In [Be12, ch. 6], Belbin elaborates different kinds of interpersonal chemistry at work places which we regard as an important factor when establishing an EA team within an organization. However, for the further discourse of the present thesis, we stick to a mere functional description of roles within Federated EA Model Management.

In [RSV08], van der Raadt et al. provide such a functional description for EA management in general. They discuss the different key EA stakeholders and group them by aspect areas as well as organizational levels. The identification of key EA stakeholders is certainly an important topic worthwhile to be mentioned in a thesis that centers around EA management. Further readings on general stakeholder management can be found in [RSV08] and [Ha10, pp. 97–102]. We abstract from the extensive list of key EA stakeholder described in [RSV08], focus on roles relevant for Federated EA Model Management, and elaborate descriptions thereof in the following.

In line with Fischer et al. [FAW07] and our research results presented in [RHM13b], we advocate that an EA team commonly consists of multiple enterprise architects whereas the modeling communities are considered as EA stakeholders (cf. [RSV08]). Figure 4.2 depicts the relationship of a federated EA model environment and Federated EA Model Management as well as the various EA stakeholders of the modeling communities and enterprise architects within the EA team. As suggested the different roles within a Federated EA Model Management initiative follow an iterative process; these efforts take place in a federated EA model environment. Subsequently, we detail further characteristics of this environment and Federated EA Model Management.

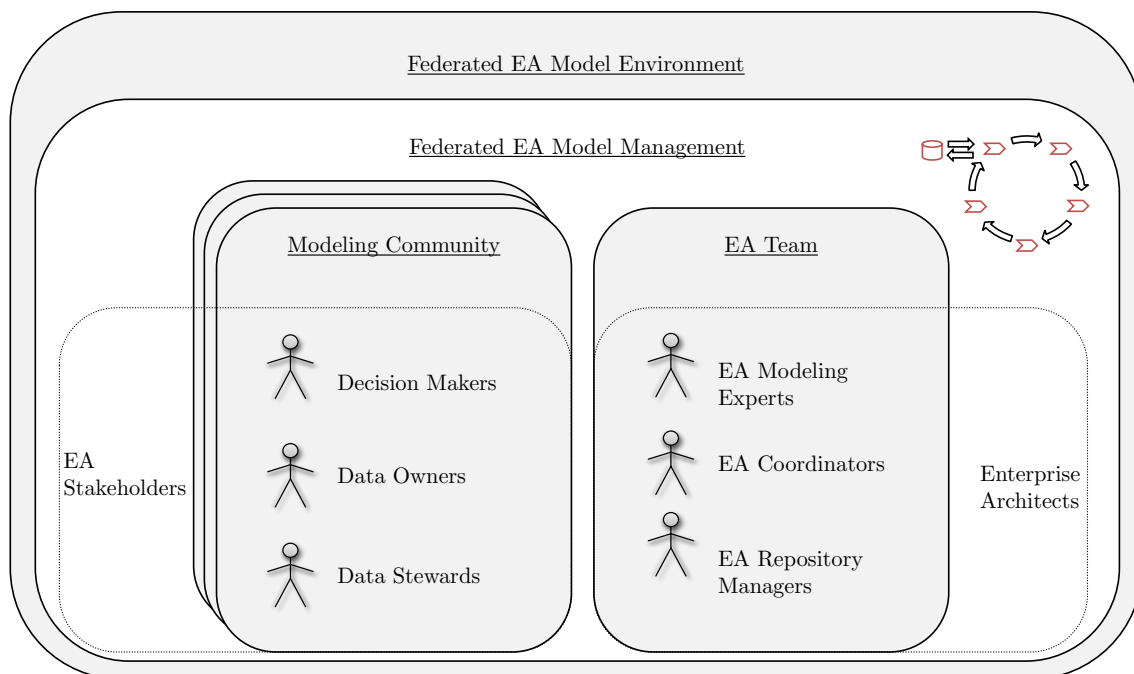


Figure 4.2: Stakeholders of a federated EA model environment: roles of the EA team and modeling communities

4.1.1.1 Roles of the EA Team

The EA team within Federated EA Model Management consists of *EA coordinators*, *EA modeling experts*, and *EA repository managers*. Within an organization, individuals possessing one or more of these roles are all known as *enterprise architects*, i.e. enterprise architects specialize in at least one of the above roles.

Enterprise architect. Hanschke describes the role of an enterprise architect as follows.

“Enterprise architects are tasked with documentation and analysis, strategic planning and control of one section of the current and future landscape model. They have overall stewardship of one part of the landscape model. When detailed data is collected from sources right across the enterprise, it will inevitably vary in terms of quality, granularity, and topicality. It is the task of the enterprise architects to pull this data together, quality-assure and consolidate it” [Ha10, p. 100]. For further readings on the topic, Op’t Land et al. [OPW⁺09, pp. 113–125] provide an extensive description of the competencies relevant for an enterprise architect.

EA coordinator. The EA coordinator is responsible for managing the maintenance process of an EA model and reports to management [RHM13b]. Also, the EA coordinator is responsible for deciding which information sources are integrated with the EA model to synchronize information in a unidirectional manner. Based on stakeholder demands it might not be useful to integrate arbitrary information sources with the EA model. The EA coordinator is tasked to estimate efforts of the procurement of information weight these against stakeholder benefits (cf. [Ha10, p. 103]). Next to determining which model is relevant to support decision makers, the EA coordinator is responsible for initiating and controlling the EA model maintenance process [FAW07].

EA repository manager. The EA repository manager is mainly responsible for technical issues and defines model mappings for the information sources that are synchronized with the EA model. Besides being responsible for further administrative tasks, the EA repository manager supports the Federated EA Modeling Community to resolve model conflicts that arise in the course of model synchronizations. The EA repository manager has a technology-oriented perspective and is capable to configure and adapt the different technical settings including complex model-to-model transformations. Thus, the EA repository manager exhibits particular expertise in a broad variety of query languages and applies these on a regular basis.

EA modeling expert. The EA modeling expert brings special expertise in the area of general modeling theory and has a particular focus on integrating different models with the EA model. An EA modeling expert is tasked to grasp a structural and semantical understanding of different models of modeling communities. Further, an EA modeling expert copes with systematic model conflicts, tries to identify patterns and develops resolution strategies to resolve conflicts efficiently. It is the EA modeling expert who ensures quality of models within the entire enterprise.

EA stakeholders can contribute information and at the same time use existing information from other sources, which they initially did not possess as a decision base. Hanschke

illustrates this relationship in [Ha10, p.99]: especially project portfolio managers and project managers can be ‘*data providers*’ and ‘*beneficiaries*’, i.e. ordinary EA stakeholders, at the same time. Armour et al. also discuss the different perspectives EA stakeholders take [AKL99a, AK01a].

“*A database administrator, for example, may focus on the structure and location of specific databases, while a sales executive may be focused on the location and movement of data through multiple information systems. Business executives are often more interested in seeing how information flows through the organization and which high-level information systems and applications support the business operations.*” [KAV05]

In the light of this diversity, we continue with a description of roles EA stakeholders engage in during the course of Federated EA Model Management.

4.1.1.2 Roles of the Modeling Communities

Although the description of EA stakeholders and respective roles is rather general in literature, we synthesized specific functional roles relevant for Federated EA Model Management. These are *decision maker*, *data owner*, and *data steward*.

EA stakeholder. EA stakeholders can be part of IT as well as business units in the enterprise and serve as first contact person between their organizational unit and the EA team. They can provide information on the current state of the EA that is not available in the existing information sources in the enterprise.

Decision maker. Decision makers play an important role for EA management endeavors. In a Federated EA Model Management, their primary intent is to consume information, i.e. they immediately benefit from a coherent decision base. They may be responsible and accountable for information sources whether this is an information system or human. Thus, they can influence and support the maintenance of an EA model.

Data owner. Data owners are experts for the repository of their respective modeling community. The data owner knows all the concepts of an information source’s metamodel, their semantics including attributes and relationships among concepts. Data owners assist the EA team to specify and maintain the interfaces to the EA model. They can resolve conflicts that result from a synchronization of the information source of their community with the EA repository—they are in possession of domain knowledge and participate in the day-to-day business of their modeling community.

Data steward. Data stewards contribute information to the EA model. Hanschke calls this role ‘*data provider*’ [Ha10, p.101]. Unlike enterprise architects who have a holistic perspective on an enterprise, i.e. are commonly interested in aggregated information [WF06], data stewards provide detailed information [Ha10, p.101].

We conclude that members of modeling communities within a federated EA model environment embrace EA stakeholders and enterprise architects. Team members may specialize

and possess different roles within Federated EA Model Management. Depending on the modeling community and task technology fit, the repositories used to support work vary. In the different modeling communities there is a considerable difference of these special purpose repositories. In the next section, we take a closer look at these repositories.

4.1.2 Repositories of Modeling Communities

As outlined in Section 4.1, information contained in multiple special purpose repositories can be integrated in a single EA repository. In the following sections we describe the nature of these systems.

4.1.2.1 Special Purpose Repositories

In a joined research initiative with University of Innsbruck, we carried out an empirical study analyzing different information sources that are utilized for EA model maintenance [FBH⁺13]. We surveyed 123 EA practitioners and questioned which information sources they use, how they perceive the data quality of these information sources, and which difficulties they were faced with during the integration thereof. We analyzed the following categories of information sources:

- Infrastructure and network monitors and scanners,
- CMDB solutions,
- PPM solutions,
- Enterprise Service Bus (ESB) solutions,
- Change management solutions, and
- License management solutions.

Figure 4.3 gives an impression of the frequency these information sources are actually utilized in practice. A comprehensive description of the related data set is provided in [FBH⁺13]. For these information sources, we analyzed the following information quality attributes:

- **Actuality**, i.e. is the information contained up-to-date?
- **Completeness**, i.e. is the information maintained (not null)?
- **Correctness**, i.e. does the information reflect the state of the real-world?
- **Granularity**, i.e. is the granularity of the information appropriate in the EA context?

The EA experts rated the different criteria according to their attitude on a five-point Likert scale [Li32] from 1 (very bad) to 5 (very good). Additionally, they could just indicate

4. Requirements Analysis

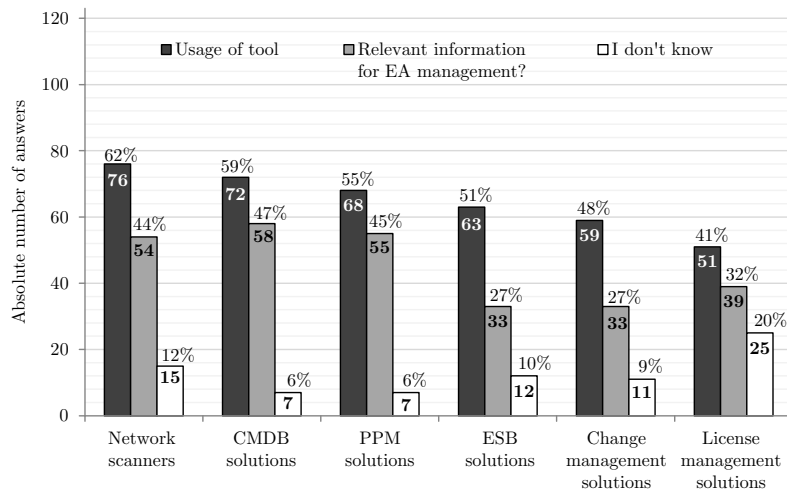


Figure 4.3: System usage and relevance as EA information sources (n=123) [FBH⁺13]

that they employ such an information source to procure information for EA management without judging the data quality. This is denoted by a dash ('-'). Figure 4.4 illustrates the different information quality criteria as they are perceived by EA experts. Since not all of the 123 EA experts have integrated the same information source with their EA model, the total number of answers varies and is therefore given in the caption of the respective figure.

We conclude that information sources maintained manually are perceived as less reliable (cf. Figure 4.4). In contrast, information systems that reflect the actual operative environment are perceived as a more reliable source, cf. Network scanners (Figure 4.4(a)) and Enterprise Service Bus (ESB) solutions (Figure 4.4(d)). For additional information on the data quality of a particular ESB solution that is frequently used in industry, we refer the interested reader to Grunow et al. [GMR12] and Farwick et al. [FBH⁺13].

Figure 4.5 depicts a classification scheme of information sources relevant for Federated EA Model Management. These range from plain text contained in documents over spreadsheets to fully blown business applications employing sophisticated ontologies. That means an information source must not necessarily feature an explicit metamodel². For instance, natural language lacks any structure that can be analyzed by an information system using common techniques³. Semi-structured information uses an explicit reference to a unique classifier. However, semi-structured information is not typed and to a large extent mixed with unstructured content [Ne12, p. 19]. Structured information on the other hand has additional constraints that makes information accessible to machines by a defined structure. These constraints range from value constraints [HM10, p. 220ff] sometimes also called object type constraint or data type constraint [Ne12, p. 33], over occurrence frequencies [HM10, p. 278ff], to more sophisticated mechanisms like ring constraints [HM10, p. 283ff]. In contrast to unstructured information, for structured information a wide range of analytics is available (see e.g. [MZ08, p. 41ff]).

²We refer the interested reader to [BMR⁺10a] for a discussion on implicit, so-called mental models, and explicit models in EA design.

³We do not consider Natural Language Processing (NLP) techniques here. For a comprehensive description about the advantages and limitations of NLP, see [ID10].

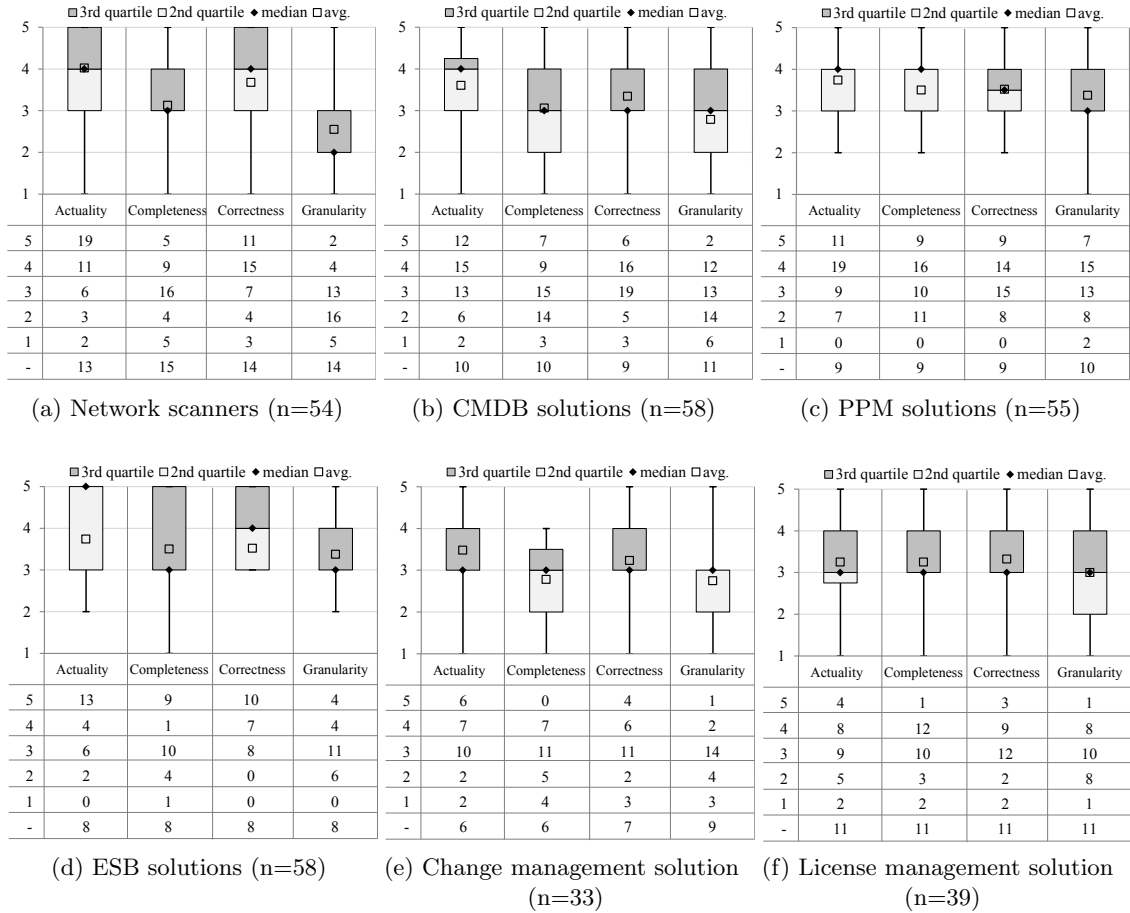


Figure 4.4: Empirical results on information quality attributes of different information sources [FBH⁺13]

Another class of information systems uses ontologies to describe their concepts. In this context the schema is most-often called domain ontology. In contrast to a schema in the classical sense, ontologies seek to define semantics of their concepts formally. “Ontologies generally appear as a taxonomic tree of conceptualizations, from very general and domain-independent at the top levels to increasingly domain-specific further down in the hierarchy” [CJB99]. This is accomplished by deriving ‘*domain ontologies*’ from an ‘*upper ontology*’⁴ [NP01]. With respect to a federated EA model environment, the former concept would define local specifics of a modeling community whereas the latter defines globally accepted semantics of more general types, i.e. could serve as a federated EA model. Ontologies pose a well-founded means to cope with mapping problems [HG01], [SS09, p. 573ff] and respective tool support moreover facilitates reasoning and type inference. From a mere technical perspective, the representation of an ontology is commonly (more) homogeneous. Two prominent examples for ontology representations are RDF [W304a], [SS09, p. 71ff] and Web Ontology Language (OWL) [W312], [SS09, p. 91ff]. However, interactions between ontologies pose a considerable challenge for researchers [SS09, p. 293ff].

⁴An upper ontology sometimes is also called top-level ontology [CJB99], [SS09, p. 279].

4. Requirements Analysis

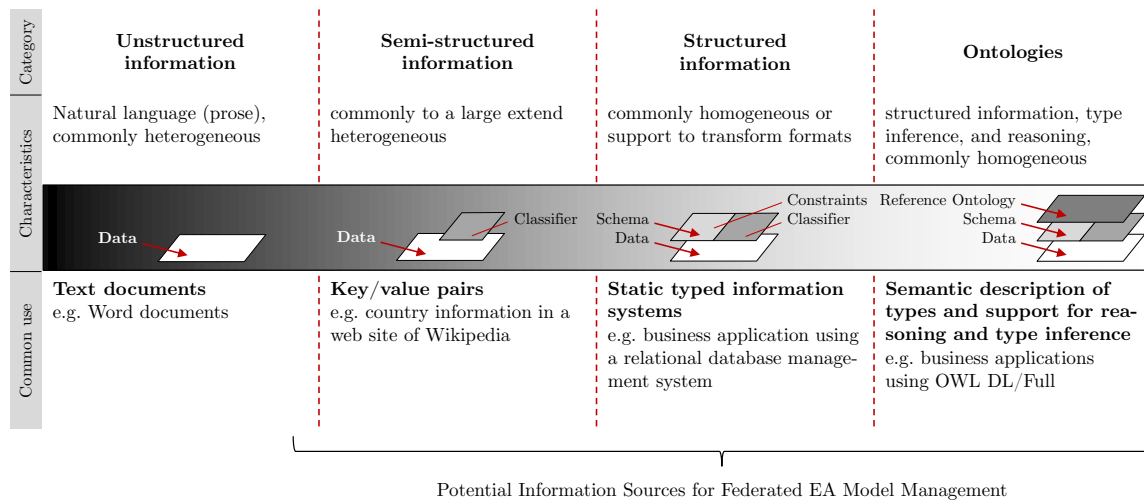


Figure 4.5: Classification of different information sources within an enterprise

The creation process of an ‘ontology’ is called ‘ontology engineering’. Creating a domain ontology commonly follows a top down approach, i.e. a domain ontology is derived from an existing upper ontology [NP01]. We claim that ontologies are not particularly suited for Federated EA Model Management for different reasons. First, in Assumption 4.2, we stated that we consider legacy environments. Respective legacy systems do not feature ontology support as the adoption of ontology models for knowledge management still ongoing [MMS⁺03],[SS09, pp. 713–734]. Second, ontology engineering takes particularly more effort, cf. e.g. [SMJ02], and new problems arise [NK04]. Third, ontology models are richer in their expressive power [De02, NK04]. During our expert interviews (cf. Chapter 7) we found that EA practitioners have a hard time to find ontology engineers within their organization and, thus, consider ontologies as a cutting edge technology that is mostly used in an academic setting. Fourth, even if ontologies are used to describe concepts, they can be transformed to the basic object-oriented models, e.g. [FSS03],[KR07, p. 229].

Revisiting Figure 4.1, repositories of different modeling communities may serve as information source for the EA repository. Commonly these information sources capture information in a structured manner conforming to well-defined concepts. Depending on the information source, their metamodel is either stored within the information source or can be reverse engineered with the help of an expert for a particular information source. Considering the multitude of potential information sources within an enterprise, we concentrate on core object-oriented modeling facilities in Assumption 4.5, since the common denominator between different systems is often made up of the concepts: objects, attributes, and relationships between objects, cf. [SL90] or [HK87].

ASSUMPTION 4.5: Information source

We assume a model of an information source always conforms to a metamodel. This metamodel may exist implicitly or explicitly. It can be either specified using an object-oriented modeling paradigm or can be mapped to an existing object-oriented model and metamodel.

In addition, many authors cope with problems that arise transforming from different paradigms to object-oriented paradigms, e.g. Fong [Fo97] describes a methodology to convert relational to object-oriented databases.

4.1.2.2 EA Repositories

Matthes et al. describe three types of EA repositories [MBL⁺08, pp. 344–345]. We summarize these types below.

metamodel driven tools feature mechanisms for metamodel adaptations. Their capabilities vary; some tools adhere to standardized metamodeling facilities, e.g. the MOF, while other tools are equipped with proprietary facilities. These may exert limitations on the possible metamodel adaptations.

methodology driven tools provide a comprehensive predefined metamodel. These tools allow only minor adaptations of the underlying metamodel, e.g. the introduction of new attributes, but commonly do not allow extensive structural changes.

process driven tools give maximum guidance for EA management. Matthes et al. describe the process driven approach as an extension to the methodology driven tools, complementing them with defined workflows that guides an EA management initiative by describing concrete activities of predefined roles.

Federated EA Model Management can be considered an iterative and incremental approach to EA model maintenance and thus the metamodel of an EA repository must be adaptable (cf. Assumption 4.6). We conclude that methodology and process driven approaches are too rigid and inflexible to support Federated EA Model Management.

ASSUMPTION 4.6: EA repository

We assume an EA repository that adheres to a metamodel driven approach and it is able to serve as a meta-modeling platform. In particular, we assume the EA repository features mechanisms which allow

- to create and alter models,
- to create and alter object definitions (classes), and
- to create and alter attribute definitions

within the EA repository.

In [RZM14], we analyze the state-of-the-art in EA tools. Thereby, we put particular focus on visualization capabilities as well as adaptability of the metamodel. In the study, 19 tools of 18 vendors have been investigated. We refer the interested reader to this study that details which EA tools support such adaptations.

4.1.3 Models within a Federated EA Model Environment

Models within an EA repository and information sources share a special relationship. In the following, we detail our observations on this relationship⁵.

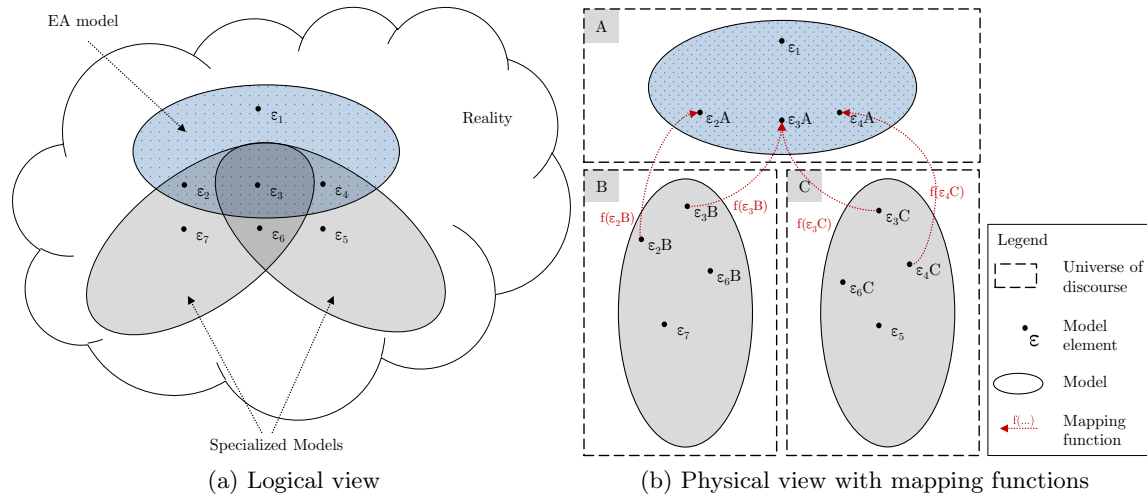


Figure 4.6: Semantic relationship between an EA model and specialized models

Figure 4.6(a) illustrates semantic relationships between three different models. Each model seeks to describe a certain part of reality. This described part of the reality is referred to as a ‘*universe of discourse*’ [Bo53, pp. 30–31]. Different universes as well as the models used to describe them may overlap semantically. Depending on the chosen universes this overlap may look differently. The illustration shows the semantic relationship between an EA model and two specialized models, which can be considered models of information sources. In an enterprise, different communities use highly specialized models to accomplish their tasks whereas an EA model serves as the glue between these specialized models. In [RM14] and Section 2.1, we emphasize the importance of information on the relationships between different real-world objects for EA management. To explain the relationship between the different models, we introduce the notion of a model element.

DEFINITION 4.4: Model element

A model element is an element in a model that describes or conceptualizes real-world objects or properties thereof. It features either an explicit identity or an implicit identity through membership in another model element. In contrast to identity of a model element, we call the property of the model element that reveals its kind *classifier*, e.g. OBJECT or ATTRIBUTE are classifiers of model elements. ■

The EA model contains elements that seek to describe aspects beyond the scope of other, specialized models, cf. \mathcal{E}_1 in Figure 4.6(a). However, it also contains elements from specialized models, cf. $\mathcal{E}_{2,3,4}$. The specialized models embrace model elements that are used to

⁵ With respect to multidatabase systems, Sheth and Kashyap [SK93] observed similar relationships.

accomplish particular tasks and, thus, may be unique with respect to other models, cf. $\mathcal{E}_{7,5}$. A specialized model may

- overlap with the EA model, cf. $\mathcal{E}_{2,4}$,
- overlap with another specialized model, cf. \mathcal{E}_6 , or
- overlap with one or more specialized model(s) and overlap with the EA model, cf. \mathcal{E}_3 .

In the following we examine these relationships more closely. Figure 4.6(b) illustrates the models viewed separately according to their universe of discourse denoted **A**, **B**, and **C**. A special notion is put on the mapping function $f : \mathcal{M} \rightarrow \mathcal{M}$ which describes complex model-to-model transformations possibly involving human intervention adding further information. This function is subject of our later analyses (cf. Section 4.2.3 and Section 5.2). For now, we concentrate on the semantic relationships. We assume that universe \mathbb{U}_A is EA management (**A** in Figure 4.6(b)), \mathbb{U}_B is PPM (**B** in Figure 4.6(b)), and \mathbb{U}_C is ITSM (**C** in Figure 4.6(b)). Models $\mathcal{M}_{A,B,C}$ describe the respective universe using model elements $\mathcal{E}_{1..n}A, \mathcal{E}_{1..n}B, \mathcal{E}_{1..n}C$. In the following, we give a more formal description followed by a concrete example for the model phenomena observed.

Our first observation concerns model elements whose semantics can be shared by two different universes.

$$\mathcal{E}_2B \mapsto \mathcal{E}_2A \wedge \forall \mathcal{E}_xC \in \mathcal{M}_C (\mathcal{E}_xC \not\mapsto \mathcal{E}_2A) \quad (4.1)$$

In Example 4.4 COSTS are maintained by both, EA management and PPM. PPM is interested in COSTS to estimate the overall COSTS of the project portfolio while EA management is concerned to decrease costs while increasing flexibility of the application landscape. Although ITSM is interested in COSTS, in our example they use their CMDB to manage operations only. As a consequence, the concept COST is currently not captured by their model.

A similar relationship can be observed between EA management and ITSM.

$$\mathcal{E}_4C \mapsto \mathcal{E}_4A \wedge \forall \mathcal{E}_xB \in \mathcal{M}_B (\mathcal{E}_xB \not\mapsto \mathcal{E}_4A) \quad (4.2)$$

In Example 4.4 OPERATING SYSTEMS are maintained by ITSM to be able to react to security vulnerabilities adequately. EA management also keeps track of the OPERATING SYSTEMS to make platform decisions in a more strategical manner with respect to BUSINESS APPLICATIONS currently running and planned to be build in the future.

Our next observation is concerned about model elements that are shared not only by two models.

$$\mathcal{E}_3B \mapsto \mathcal{E}_3A \wedge \mathcal{E}_3C \mapsto \mathcal{E}_3A \quad (4.3)$$

In Example 4.4 BUSINESS APPLICATIONS are of high interest for PPM, EA management as well as ITSM; each from a slightly different angle. While PPM is concerned to carry out projects, ITSM is interested in BUSINESS APPLICATIONS to be able to contact the respective owner prior to unexpected maintenance cycles. EA management takes a holistic perspective and seeks to analyze the interconnections between BUSINESS APPLICATIONS.

While in the above outlined examples EA management had an interest in the model elements shared, there may be semantic relationships between model elements that are not (yet) of interest for EA management.

$$\mathcal{E}_6 B \mapsto \mathcal{E}_6 C \wedge \forall \mathcal{E}_x A \in \mathcal{M}_A (\mathcal{E}_6 B \not\mapsto \mathcal{E}_x A \wedge \mathcal{E}_6 C \not\mapsto \mathcal{E}_x A) \quad (4.4)$$

In Example 4.4 the communities could share information about SLAs. PPM is concerned about requested SLAs and COSTS associated with the PROJECTS while the SLAs have to be met by ITSM.

Our final observations address model elements that are maintained solely in one universe.

$$\exists \mathcal{E}_7 \wedge \forall \mathcal{E}_x \in \mathcal{M}_{A,C} (\mathcal{E}_7 \not\mapsto \mathcal{E}_x) \quad (4.5)$$

In Example 4.4 PROJECTS are only subject of interest for PPM.

$$\exists \mathcal{E}_5 \wedge \forall \mathcal{E}_x \in \mathcal{M}_{A,B} (\mathcal{E}_5 \not\mapsto \mathcal{E}_x) \quad (4.6)$$

In Example 4.4 SERVERS are only subject of interest for ITSM.

$$\exists \mathcal{E}_1 \wedge \forall \mathcal{E}_x \in \mathcal{M}_{B,C} (\mathcal{E}_x \not\mapsto \mathcal{E}_1) \quad (4.7)$$

In Example 4.4 INFORMATION FLOW is only subject of interest for EA management.



EXAMPLE 4.4: Overlapping models within an enterprise

Figure 4.7 shows a minimalistic example of models from EAM, PPM, and ITSM respective entities and their semantic relationships. In order to derive planned states of an EA any of these entities are subject of interest for EA management. However, currently EA management focuses on the BUSINESS APPLICATIONS and the INFORMATION FLOW between them, their OPERATING SYSTEM and involved COSTS. PPM in contrast concentrates on PROJECTS. Also, they keep track of requested SLAs as well as involved COSTS for hosted BUSINESS APPLICATIONS. With this information PPM is able to decide on projects and to estimate COSTS based on requested SLAs. ITSM primarily is concerned to track runtime information such as hardware failure as well as inventory records. Thus, it maintains a list of all SERVERS within the enterprise as well as information about their OPERATING SYSTEMS. In order to improve their level of service, ITSM also maintains a list with SLAs they have to meet. On a maintenance cycle, ITSM has to inform the respective owner of a BUSINESS APPLICATION that may be influenced by updating the OPERATING SYSTEM or SERVER hardware.

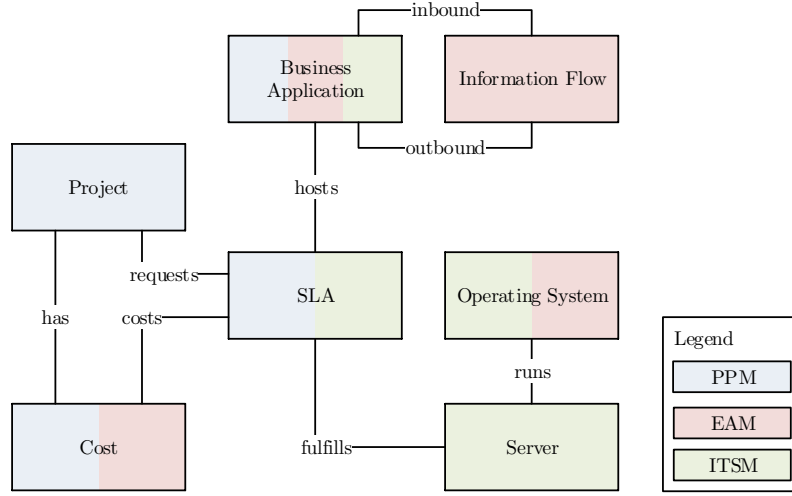


Figure 4.7: Entities and relationships of models of EAM, PPM, ITSM

Figure 4.7 takes a holistic perspective on different models describing different universes of discourse. Viewed separately, the local perspectives are restricted.

- PPM cannot tell the actual costs involved, because with the given information PPM does not know the information flow between business applications and thus cannot estimate the impact on other, currently existing, SLAs.
- ITSM cannot tell the actual impact of hardware upgrades or software updates because the INFORMATION FLOW so as the interfaces to other BUSINESS APPLICATIONS are not known.
- EA management cannot decide to homogenize the OPERATING SYSTEMS because they do not have any information of the SERVERS the OPERATING SYSTEMS are running on. Especially the future demands of projects that will be carried out are unknown to EA management.

We generalize from the example outlined above and describe a federated EA model environment formally.

DEFINITION 4.5: Federated EA model environment

Model $\mathcal{M}_1 \in \mathbb{U}_1$ builds a *federal model* with the *federated models* $\mathcal{M}_{2..n} \in \mathbb{U}_{2..n}$ iff:

$$\mathcal{M}_{2..n} \overset{\circ}{\mapsto} \mathcal{M}_1 \Leftrightarrow \forall \mathcal{M}_{2..n} \exists \mathcal{E}_\sigma^i \in \mathcal{M}_{2..n} (\mathcal{E}_\sigma^i \mapsto \mathcal{E}_\phi^1 \in \mathcal{M}_1) \quad (4.8)$$

If Equation 4.8 holds, $\mathcal{M}_{1..n}$ build a federated EA model environment. ■

Thereby, $\mathcal{M}_{2..n}$ as well as \mathcal{M}_1 may contain model elements not captured by models of other universes. We call these isolated model elements.

DEFINITION 4.6: Isolated model elements

In a federated model environment $\mathcal{M}_{1..n}$ with $\mathcal{M}_{2..n} \xrightarrow{\circ} \mathcal{M}_1$, model $\mathcal{M}_\sigma \in \mathcal{M}_{1..n}$ has isolated model elements iff:

$$\exists \mathcal{E}_\sigma \in \mathcal{M}_\sigma (\nexists \mathcal{E}_\phi \in \mathcal{M}_{1..n} \setminus \mathcal{M}_\sigma (\mathcal{E}_\sigma \mapsto \mathcal{E}_\phi \vee \mathcal{E}_\phi \mapsto \mathcal{E}_\sigma)) \quad (4.9)$$

■

Isolated elements are maintained by a *modeling community* of one universe of discourse. Such a modeling community can also be considered a ‘*linguistic community*’. A linguistic community speaks the same language and, thus, members of a linguistic community share a common understanding, i.e. mental model, when using specific terms to describe real-world objects [BMR⁺10a]. Sometimes the universes described by the communities overlap and a cross community understanding is formed.

In contrast to isolated elements, we introduce the notion of *boundary model elements*. Boundary elements are subjects of interest to multiple universes of discourse and respective linguistic communities. The concept of boundary model elements is derived from the notion of ‘*boundary objects*’; a term initially coined by Star and Griesemer [SG89]. In their paper, the authors point out the “problem of common representation in diverse intersecting social worlds” [SG89]. In the context of Zoology and in a broader sense scientific work, Star and Griesemer [SG89] not only describe the central role of a repository, but also the necessity to build a shared understanding of ‘*boundary objects*’ among groups with diverse backgrounds. These intersecting, heterogeneous groups face diversity and the challenge to ensure consistency in information in order to solve scientific problems. This poses especially a challenge when exchanging information that has different meanings in different universes. The authors state that the “*n*-way nature of the [...information exchange...] cannot be understood from a single viewpoint” [SG89] and outline social implications. In their study, diverse parties that share a common goal participate concurrently in heterogeneous work. Star and Griesemer [SG89] also emphasize the importance of autonomy and communication between different ‘worlds’, i.e. universes. The authors confirm that some information are interesting for all these ‘*social worlds*’ and describe two ways to overcome challenges, i.e. methods standardization and the development of boundary objects.

“Boundary objects are objects which are both plastic enough to adapt to local needs and the constraints of the several parties employing them, yet robust enough to maintain a common identity across sites. They are weakly structured in common use, and become strongly structured in individual-site use. These objects may be abstract or concrete. They have different meanings in different social worlds but their structure is common enough to more than one world to make them recognizable, a means of translation. The creation and management of boundary objects is a key process in developing and maintaining coherence across intersecting social worlds.” [SG89]

Inspired by the idea of boundary objects we include objects, attributes, relationships, and (subsets of) models and introduce the notion of boundary model elements.

DEFINITION 4.7: Boundary model elements

In a federated model environment $\mathcal{M}_{1..n}$ with $\mathcal{M}_{2..n} \xrightarrow{\circ} \mathcal{M}_1$, model $\mathcal{M}_\sigma \in \mathcal{M}_{1..n}$ has boundary model elements iff:

$$\exists \mathcal{E}_\sigma \in \mathcal{M}_\sigma \left(\exists \mathcal{M}_\phi \in \mathcal{M}_{1..n} (\exists \mathcal{E}_\phi \in \mathcal{M}_\phi (\mathcal{E}_\phi \mapsto \mathcal{E}_\sigma \vee \mathcal{E}_\sigma \mapsto \mathcal{E}_\phi \wedge \mathcal{M}_\phi \not\equiv \mathcal{M}_\sigma)) \right) \quad (4.10)$$

■

Referring to the work of Star and Griesemer, Wenger [We98, pp. 106ff] notes that not every object is a boundary objects (cf. Definition 4.6). Wenger further adds that perspectives may change and an object may become a boundary object. Aier et al. [AKS⁺08] state that an EA model captures information in a rather broad and aggregated form. The authors state that an EA model should not include concepts that do not influence the EA. In line with these observations, we advocate that an EA model as well as its metamodel evolve over time, cf. [RHM13a]. Typically, relationships add most value in an EA model for decision makers (cf. Chapter 7 and Section 2.1).

4.1.4 Consistency within a Federated EA Model Environment

Next to our analysis of the semantic relationships among models, we detail different notions of consistency that exist within a federated EA model environment.

In DBMS different notions of consistency exist. Date [Da04, p. 264] lays grounds that let us conclude a consistent system does not necessarily lead to a sound decision base. “The system cannot enforce truth, only consistency” [Da04, p. 264]. Date further continues to clarify that correctness implies consistency but consistency does not imply correctness [Da04, p. 265]. In addition, inconsistencies imply incorrectness and incorrectness does not imply inconsistencies. This is a very strict perspective and certainly meaningful from a DBMS perspective. In this thesis we take a less strict position. While data might be inconsistent, humans can still retrieve the relevant information from it. We argue that for a knowledge worker, correct information is sufficient to carry out tasks [Da05]. For automated calculations on the other hand, correct data is required, e.g. in order to aggregate values [DHM10, p. 31]. In line with Date, by *correctness* or *correct*, we understand information that “reflects the true state of affairs in the real world” [Da04, p. 265].

**EXAMPLE 4.5: Data consistency vs. information consistency**

The number of licenses purchased in the license management solution displays the integer ‘5’ for Customer Relationship Management (CRM) solutions and the string ‘five’ for Enterprise Resource Planning (ERP) solutions.



Example 4.5 illustrates a case in which a knowledge worker can decide; the string ‘five’ does have the same meaning to a human as the integer ‘5’ in many contexts. For now, we close

the discussion with the conclusion that an information system can function for a particular purpose of knowledge workers even if there are inconsistencies. In the remainder of this thesis we continue the discussion about the *necessary degree of freedom* in a federated EA model environment.

With regards to distributed DBMSs Vogels reports on ‘*eventual consistency*’ in [Vo08, Vo09]. The Chief Technology Officer (CTO) and Vice President of Amazon.com presents different notions of consistency for distributed systems.

Strong consistency means that on an update and subsequent access, a system returns the updated value right after the update completes.

Weak consistency gives no guarantees to subsequent read access. A set of defined conditions must be met before the updated value is returned. Vogels calls the period between the initial update and the moment when the system can guarantee that any subsequent access will return the updated value ‘*inconsistency window*’.

Eventual consistency is a specific form of weak consistency; it is guaranteed that if no further updates are made to the object updated initially, the system will return the last updated value eventually. “If no failures occur, the maximum size of the inconsistency window can be determined based on factors such as communication delays, the load on the system, and the number of replicas involved in the replication scheme” [Vo09].

Building on the different notions of consistency in distributed systems, we establish an understanding of terms that denote consistency in a federated EA model environment. We start with consistency within a single information source.

DEFINITION 4.8: Local consistency

Local consistency denotes the conformance of a federated information source’s model and its elements to its metamodel. ■

Each information source can implement a consistency paradigm on its own, e.g. one information source could implement strong consistency whereas the next could favor weak consistency and so on. In the remainder of this chapter, we outline that these potentially heterogeneous information sources are unidirectionally synchronized with a federal EA repository.

DEFINITION 4.9: Federal consistency

Federal consistency denotes the conformance of the EA repository’s model to its metamodel, i.e. local consistency of the EA repository. ■

Our final notion of consistency is concerned with the entire federation. Contradicting information could be misleading for decision makers or fracture trust in the EA repository.

PRELIMINARY DEFINITION 4.1: Cross-community model consistency
Cross-Community Model Consistency (CCMC) or *global consistency* denotes the semantic integrity of all information sources with the EA repository and among each other. □

Assumption 4.7 details that Cross-Community Model Consistency (CCMC) is a goal constantly shared by all members of the modeling communities. However, there are exceptions that aggravate to reach CCMC or even prohibit to reach it. We further detail these exceptions and the different states of a model element in Chapter 5.

ASSUMPTION 4.7: Organizational culture
 CCMC is a vision shared among all modeling communities. The different community teams constantly work towards CCMC.

4.2 Use Case Analysis of Federated EA Model Management

After giving an overview of Federated EA Model Management and its characteristics with respect to involved roles, repositories, and models, we proceed to analyze the use cases of different stakeholders. These use cases are derived from our research experience [BEG⁺12, HMR12, RHM13b, RHF⁺13, FBH⁺13, RHM13a] and have been validated in the course of the expert interviews (cf. Chapter 7). They build the foundation of the requirements we determine for our approach to Federated EA Model Management and its software support, namely MODELGLUE. Prior to discussing details, Figure 4.8 gives an overview of the use cases.

The use cases are split in manual use cases and use cases that address three information systems, namely a federal system, an EA repository, and an information source. The manual use cases do not involve an information system. They are however prerequisite and serve to prepare essential artifacts that are utilized during other use cases. We give a comprehensive description for the manual use cases as well as for use cases that concern the federal system. For the remaining use cases, we provide a brief overview and emphasize their relationship to other use cases that address the federal system (cf. Section 4.2.13).

In the remainder of this section, we motivate essential parts of the use cases briefly, provide an overview of the solution space as well as pointers to literature and describe each of the use case for a federal system in a tabular manner. Thereby, we employ a structure that can be used as a template for use case analysis.

4.2.1 A Template for the Structured Presentation of Use Cases

In [Co01, ch. 11], Cockburn discusses different use case templates. Although he favors the ‘*fully dressed use case*’, he also introduces the ‘*casual*’ template which omits certain sections as well as more structured variations, namely the one-column table and two-column table template. Cockburn states that “others often choose the table style” [Co01, p. 122]. For

4. Requirements Analysis

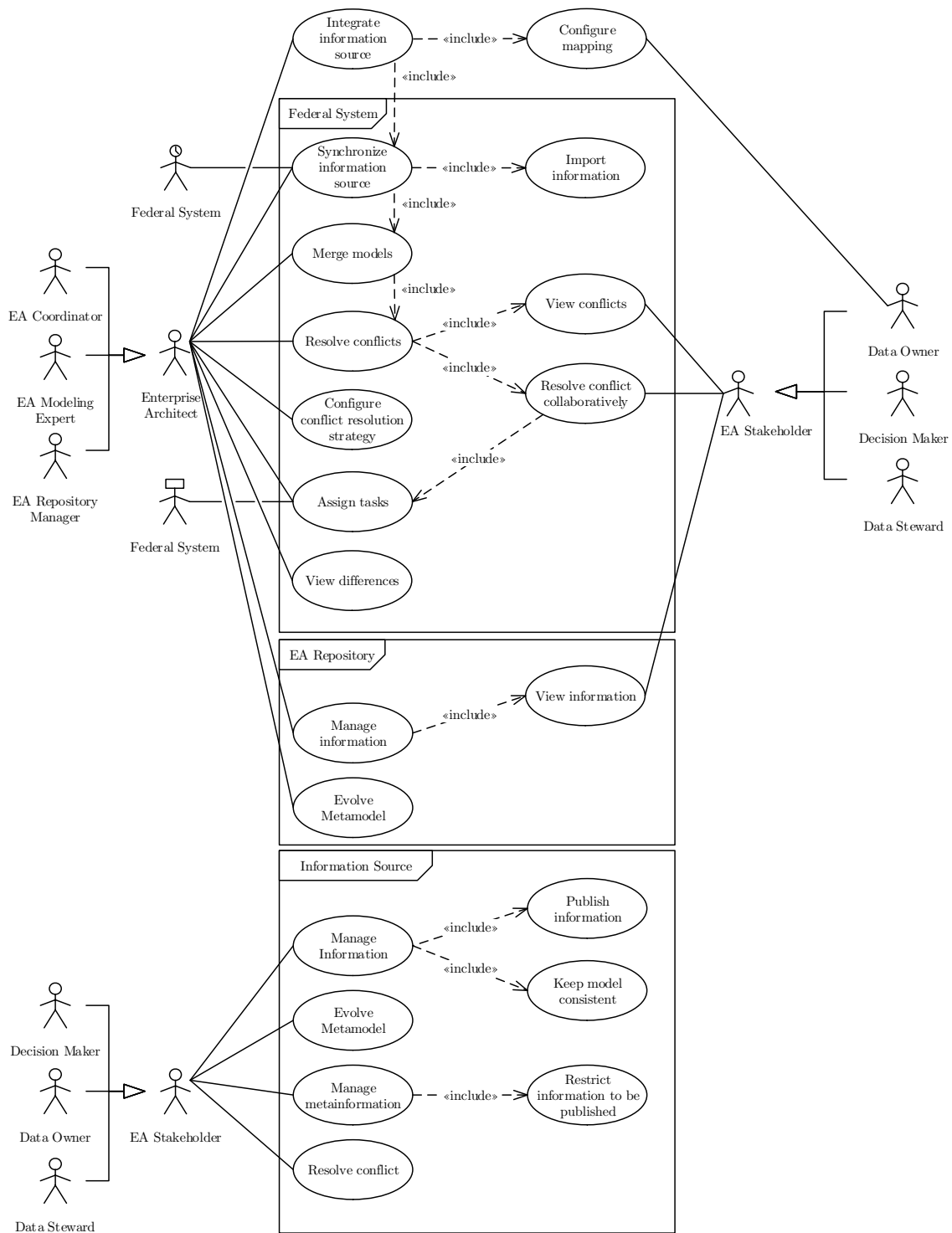


Figure 4.8: Use Cases of Federated EA Model Management

Table 4.1 illustrates the template employed to describe use cases in the present thesis. It is a slightly modified version of Cockburn’s one-column table template [Co01, ch. 11] and embraces a brief identifier, name of the use case, context, scope and level as well as main stakeholders and respective interests. The template further describes any preconditions, minimal guarantees that are fulfilled in any case and success guarantees. A trigger starts the main success scenario which describes the common flow of events using steps as well as sub steps. Extensions to this main success scenario are introduced using a branch criteria and either an immediate action or a label that refers to another use case. The final part of the template describes variations that are influenced by technology and data, e.g. it may be a difference if users access an application via a mobile device or not.

Table 4.1: Use Case Template of Cockburn [Co01, ch. 11]

Use case #	a unique identifier	Use Case Name	the name is the goal as a short active verb phrase
Context of use	a longer statement of the context of use if needed		
Scope	what system is being considered black box under design		
Level	one of: Summary, Primary Task, Subfunction		
Primary actor	a role name for the primary actor, or description		
Stakeholder & Interests	Stakeholder	Interest	
	Stakeholder	Interest the stakeholder has concerning this use case.	
Preconditions	what we expect is already the state of the world		
Minimal Guarantees	the interests as protected on any exit		
Success Guarantees	the interests as satisfied on successful ending		
Trigger	the action upon the system that starts the use case		
Main Success Scenario	Step	Action	
	1	put here the steps of the scenario from trigger to goal delivery, and any cleanup after	
	
	n	...	
Extensions	Step	Branching Action	
	1	condition causing branching : action or name of sub-use case	
	
	n	...	
Technology & Data Variations	Step	Description	
	1	variations that occur because of a technology involved	
	
	n	...	

4.2.2 Integrate Information Source

Table 4.2 presents the first use case (UC1). It addresses the initial integration of an information source with a federated EA model environment. The integration is necessary to synchronize information that is contained in an information source with information contained in an EA model. The use case includes the initial setup of an *import metamodel* that uses the terminology of the EA model. This import metamodel can be explained by employing the reference architecture for federated information systems by Sheth and Larson [SL90].

In the domain of FDBSs the concepts of an ‘*export schema*’ and an ‘*import schema*’ come close to what the data owner and the EA modeling expert intend to create in UC1. Inspired by the concepts presented first by Heimbinger and McLeod [HM85] and Sheth and Larson [SL90] (cf. Section 2.4), we define two terms central for the further discourse of this thesis.

DEFINITION 4.10: Export metamodel

An export metamodel describes the parts of a metamodel of an information source which are intended to be shared with the federation in an explicit manner. The author of the export metamodel uses the structure and terminology of the information source to describe concepts of the export metamodel and their relationships to each other. ■

Definition 4.10 transfers the idea of the ‘*export schema*’ and comes intentionally close to the notion of Sheth and Larson [SL90] and Heimbinger and McLeod [HM85].

DEFINITION 4.11: Import metamodel

An import metamodel describes parts of an information source’s metamodel which are intended to be integrated with the EA model. The author of the import metamodel uses the structure and terminology of the EA model to describe concepts of the import metamodel and their relationships to each other, i.e. an import metamodel commonly describes a subset of the EA repository’s metamodel. ■

Definition 4.11 on the other hand specifies a concept that must not be confused with the ‘*import schema*’ of Heimbinger and McLeod [HM85], i.e. an import metamodel is not specified within or for the information demand of an information source, especially since in Federated EA Model Management information sources do not import any information automatically. However, we transfer the core idea to Federated EA Model Management. In contrast to FDBS, import metamodels may represent concepts in a transformed manner such that the resulting view, i.e. the *import model*, on the original information, i.e. the *export model*, must be considered read only.

Figure 4.9 depicts the relationship of the different models as well as the conformance to a common metamodel. Note that the export metamodel commonly is not stored in the

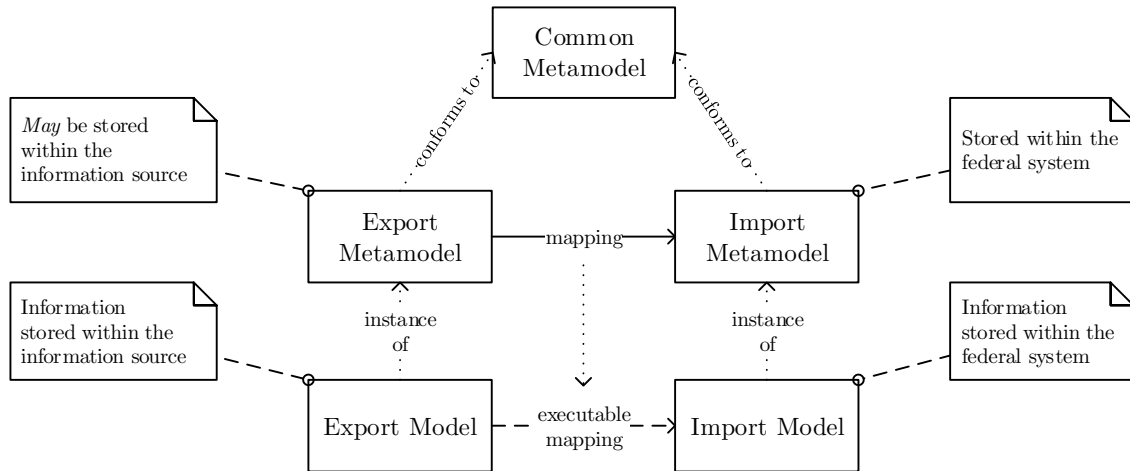


Figure 4.9: Relationship of a common metamodel, import metamodel, export metamodel, import model and export model

information source. The conformance to the *common metamodel* is similar to the Canonical or Common Data Model (CDM) of Sheth and Larson [SL90]. This overcomes heterogeneity of different metamodel languages and, thus, fosters the creation of a mapping.

After this brief excursion, we continue with the flow of activities in UC1 (cf. Table 4.2). In this use case, different parties seek to integrate an information source and prepare essential artifacts for developing a mapping between two different metamodels. Before initiating a workshop, the export metamodel of an information source is specified by the data owner. The data owner captures concepts that have been requested by the EA coordinator. This export metamodel serves as a basis for the mapping to the EA repository. In this vein, the quality of the information source’s metamodel is checked by the EA modeling expert. General model quality criteria can be found for instance in [Kr95, p. 93ff], [KLS95], and [St73, pp. 131–133]. If the EA modeling expert proposes changes, the data owner can decide to alter the metamodel according to identified potential for improvements. This step serves to prevent an uncontrolled model growth, i.e. the models and metamodels within a federation could be improved in a iterative manner potentially. In the course of a collaborative workshop semantics of concepts can be discussed and clarified. The goal is to develop a mapping between the export metamodel and an import metamodel. Once, the models and semantics of their concepts are clarified, and a mapping has been created, the EA repository manager creates the physical import metamodel within the federal system. That includes the definition of roles and access rights for the community members of the information source within the federal system.

The important steps of UC1 are to create a mapping of model elements, to refine the conflict resolution strategy, and to synchronize the information source with the EA model subsequently. Since these steps are also primary use cases that can be triggered by other events, we explain their details separately.

Table 4.2: Use Case: Integrate Information Source

Use case #	UC1	Use Case Name	Integrate Information Source
Context of use	A model of an (additional) information source should be integrated with an existing EA model.		
Scope	Federal system implementing MODELGLUE		
Level	Primary Task		
Primary actor	EA coordinator		
Stakeholder & Interests	Stakeholder	Interest	
	EA coordinator	wants to integrate information of an existing model with the EA model to utilize the contained information for decision making	
	EA modeling expert	wants to prevent and counter uncontrolled growth of models and ensure model quality through conformance to general model principles	
	Data owner	intends to share information for the common good of an organization	
	EA repository manager	has the technical skills to define the mapping and wants to increase the relevance of the EA repository as a reliable source for decision makers	
	EA stakeholders	want to have a sound decision base with consistent information	
Preconditions	An information source of interest has been chosen by the EA coordinator and all stakeholders have been informed and agree to collaborate, i.e. are involved.		
Minimal Guarantees	Any change to the EA model is logged in a version history and changes can be reverted.		
Success Guarantees	The model of an additional information source is integrated with an EA model. A unique link to the original information source is maintained for new as well as existing information.		
Trigger	The EA coordinator contacts the data owner of an information source to start this use case manually.		
Main Success Scenario	Step	Action	
	1	The EA coordinator identifies concepts within an information source, which contain information relevant for the EA team or EA stakeholders.	
	2	The data owner specifies an export metamodel that conceptualizes information to be exported; the EA coordinator formulates the information demands.	
	3	The export metamodel is reviewed by the EA modeling expert.	
	4	The EA modeling expert creates a conceptual import metamodel in collaboration with the EA coordinator.	

Main Success Scenario (cont'd)	5	The data owner, EA modeling expert, and EA repository manager collaborate to clarify semantics of the export and import metamodel in order to define mappings from the information source to an EA repository (see UC2). In this step, minor modifications to both, the export and the import metamodel are to be expected.
	6	The EA repository manager creates the physical model in the federal system based on the import metamodel provided by the EA modeling expert.
	7	The EA repository manager creates roles for new modeling community members in the federal system.
	8	The EA modeling expert and EA repository manager collaborate to refine the conflict resolution strategy (see UC9).
	9	The EA repository manager synchronizes the information source with the EA model (see UC3).
Extensions	Step	Branching Action
	9	The EA repository manager wants to view the differences to the information contained in the EA model: view model differences (see UC5)
	9	The EA repository manager just wants to import the model of an information source to the federal system rather than synchronizing it with the EA model: import information (see UC4)

4.2.3 Configure Mapping

As described above, the mapping between the export metamodel of an information source and an import metamodel of the federal system is created in one or more collaborative workshops. During these workshops, the data owner and the EA Modeling Expert create a conceptual mapping from the information source to the EA repository, which later is translated to technical model transformations. In line with Schulz, we advocate that a technical mapping is derived from an initial conceptual mapping [Sc12, p. 134]. The creation and adaptation of a mapping, i.e. its configuration, is described in the next use case, UC2, illustrated in Table 4.3.

In line with Sheth and Larson [SL90], we advocate two major reasons for a description of an export metamodel; these are

1. it describes the information to be exported in a uniform representation; hence an export metamodel overcomes heterogeneity of divergent meta-metamodels and
2. it allows adding further information that may be missing in the information source, e.g. additional semantics or type and cardinality constraints.

Table 4.3: Use Case: Configure Mapping

Use case #	UC2	Use Case Name	Configure Mapping
Context of use	A mapping between an export metamodel of an information source and an import metamodel within the federal system has to be created or altered.		
Scope	Federal system implementing MODELGLUE		
Level	Primary Task		
Primary actor	EA modeling expert		
Stakeholder & Interests	Stakeholder	Interest	
	Data owner	has domain knowledge and wants that information in the information source is understood correctly	
	EA coordinator	wants to gather information from an information source	
	EA modeling expert	wants to create or alter a mapping between an export metamodel and an import metamodel	
	EA repository manager	wants to create a technical model-to-model transformation based on the mapping created	
Preconditions	Entities and relationships within the information source are captured in an export metamodel as well as import metamodel.		
Minimal Guarantees	Only valid configurations, i.e. specifications of model-to-model transformations are stored within the federal system		
Success Guarantees	A new version of a mapping configuration for an information source is stored in the federal system.		
Trigger	The EA modeling expert coordinates with the data owner of an information source to start this use case manually.		
Main Success Scenario	Step	Action	
	1	The EA modeling expert collaborates with the data owner such that both parties understand the semantics and structure of the export metamodel and import metamodel.	
	2	The EA modeling expert collaborates with the data owner. They specify which model element of the export metamodel has to be mapped to which model element of the import metamodel; this embraces types, attributes, relationships	
	3	The EA modeling expert and the data owner specify which instances to query, i.e. they define filter criteria.	
	4	The EA Modeling expert and the data owner define access rights and a unique identifier within the information source that allows for an identity reconciliation later on.	
	5	Based on the conceptual mapping, the EA repository manager specifies transformation rules and stores these rules in a configuration file; this configuration file details which relations to materialize and how to query, i.e. traverse, the metamodels during synchronization (cf. UC3).	

Note that an import metamodel does not describe any additional transformation rules such as aggregations. These rules are part of the mapping. In the MDE community, there is a similar notion, cf. the model transformation pattern of Jouault et al. [JAB⁺08, JK06].

After this brief excursion motivating the role of an export metamodel, we continue with the description of UC2. Its ultimate goal is to provide transformation rules that detail how to traverse the model of an information source such that the relevant information can be mapped to concepts within the EA model. This includes aggregations, traversing relationships, and specifying a unique identifier for each model element imported. The latter serves as a means to identify the origin of model elements. This way, the original source, i.e. its origin, can be traced such that conflicts or incorrect values can be corrected in the information source which is leading information delivery platform for a particular model element.

DEFINITION 4.12: Identity reconciliation within a federation

Identity reconciliation is regarded as the ability to trace a particular model element to its corresponding model element in its original information source. The origin of a model element can be either an information source or the EA repository. ■

Given Definition 4.12, it is imperative that the information source is distinct within the federated EA model environment. Regarding identity reconciliation, the EA repository can be considered as an information source, too.

In line with Conrad [Co97, p. 91], we omit the description of a technical propagation of changes, i.e. a transformation from the information source to the EA model are not required to be invertible to update the model of the information source automatically. The intent of UC2 is to create a unidirectional mapping, i.e. we do not seek to propagate changes to the information source. In line with Pierce [Pi02, pp. 181–185], Schweda [Sc11, p. 179f] revisits the subsumption relationship of models with respect to read-only access and calls this ‘*weak subsumption*’ which can be viewed as a read-only embedding. Weak subsumption means automated updates are technically not feasible when a certain set of transformations is applied to a model transformation.

In a joint research cooperation with KTH Stockholm, we provide a mapping for SAP PI to ArchiMate [BEG⁺12] and illustrate an example utilizing the ATLAS Transformation Language (ATL) [JAB⁺08]. Besides this concrete example, Czarnecki and Helsen [CH03] as well as Mens and Van Gorp [MVG06] provide the interested reader with an overview of model transformation approaches.

In [HMR12], we outline that model mappings are a particular challenge for industry and various ways are required to transform one model to another. Transformation languages like ATL account for this fact. Jouault et al. state that “it is sometimes difficult to provide a complete declarative solution for a given transformational problem” [JAB⁺08]. As a solution, ATL implements a hybrid approach, i.e. it embraces declarative and imperative language constructs. “In general, model transformations may be implemented in different ways, for example, by using a general purpose programming language” [JAB⁺08]. The

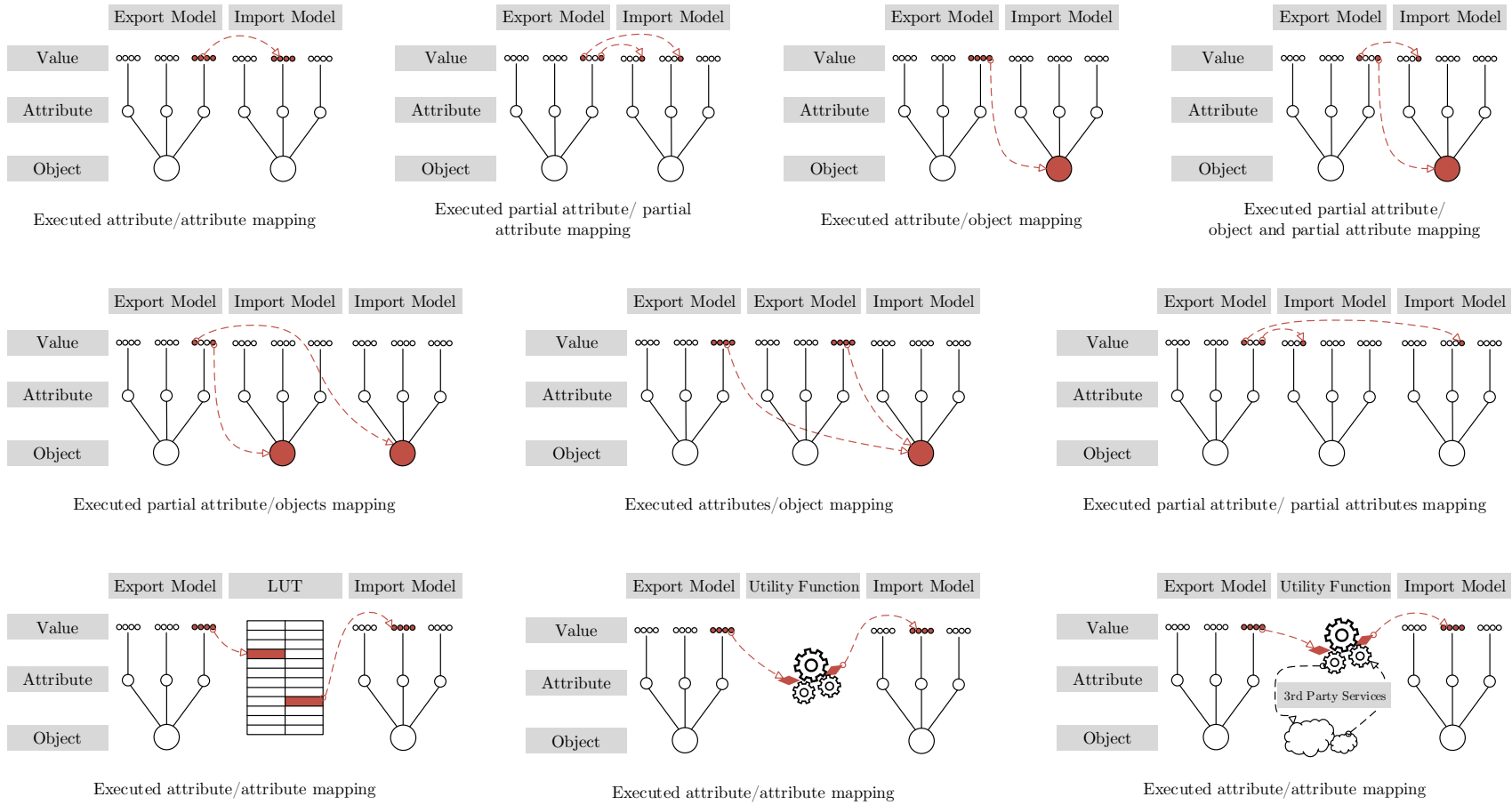


Figure 4.10: Non-exhaustive list of examples for an executed mapping of attributes of an export model to an import model

sheer variety of different mappings entails this situation. Schulz made this observation of arbitrary complex mappings in the context of large data migration projects in [Sc12, p. 129].

Figure 4.10 displays different structural transformations that can be specified with mappings for one respectively two attributes in the export model. In the simplest case, values can be transferred from one model to another by an assignment. This gets more complex when the values are split, e.g. one model contains the first and last name as one value and the other model has separate attributes for both. Values can also become entire objects. Some part of the value can be transformed to an object and any other part can become an attribute. Multiple combinations are foreseeable such as an attribute that is split and becomes two objects. Besides these split operations, also merge operations, or more complex look-ups are realistic examples for model mappings. This observation is not limited to model transformation for EA models, cf. [JAB⁺08]. For instance, a lookup table (LUT) often serves as a means to overcome synonym issues in the data cleansing process (cf. e.g. [KR02, p. 21]). In their paper, Jouault et al. [JAB⁺08] point out that a purely declarative approach is often not sensible for complex model-to-model transformations. In line with the authors we foresee utility functions that are employed to transform a value. A simple example is the transformation of a timestamp to another date format with respect to particular time zones. Further, third party services could be utilized, e.g. to look up locations based on Global Positioning System (GPS) coordinates. Note that this is a non-exhaustive list of possible mappings for one respectively two attributes and different operations can be combined in various ways.

4.2.4 Synchronize Information Source

Once, the information source is integrated in the federal system, it can be synchronized with the EA model in a unidirectional manner. This synchronization is subject of our next use case, UC3, detailed in Table 4.4. Goal of this use case is to import information of the information source with the EA model and to resolve arising model conflicts.

In the context of the extract, transform, and load (ETL) process, the DWH community outlines that different triggering events to import information exist [BG13, pp. 56,94], [KC04, ch. 3]. A general distinction can be made between pull and push. That is the information source can either push information to a federal system or vice versa, i.e. the federal system pulls information from the source.

Periodical. A synchronization takes place at a regular basis. The polling frequency strongly depends on the information change rate in the export model and required information actuality of the EA model.

Request-based. A synchronization starts upon an explicit request. This can be reasonable if a particular information source is updated at a higher frequency than usual e.g. in the course of inventory measures.

Event-based. A synchronization takes place based on an external event. These can be further classified as

- time-based events, e.g. monthly for a stakeholder report,

Table 4.4: Use Case: Synchronize Information Source

Use case #	UC3	Use Case Name	Synchronize Information Source
Context of use	An integrated model of an information source should be synchronized with an existing EA model.		
Scope	Federal system implementing MODELGLUE, information source, and EA repository		
Level	Primary Task		
Primary actor	EA repository manager		
Stakeholder & Interests	Stakeholder	Interest	
	Data owner	eventually knows reasons why models are conflicting and can resolve conflicts within the information source	
	EA repository manager	wants to synchronize a model of an information source with the EA model	
	EA stakeholder	wants to have a sound decision base with consistent information	
Preconditions	A valid configuration file to synchronize the information source exists, i.e. the model of the information source is integrated.		
Minimal Guarantees	Any changes to the import model and the EA model are tracked in a version history and can be reverted if necessary; any conflicts are detected and information about conflicts is available.		
Success Guarantees	All changes of an information source since the last synchronization are transferred to the import model stored within the federal system and are synchronized with the EA model; arising conflicts are resolved.		
Trigger	The EA repository manager coordinates with a data owner of an information source to start this use case manually or the federal system triggers this use case periodically.		
Main Success Scenario	Step	Action	
	1	The EA repository manager triggers the import of an information source manually (UC4).	
	2	After the import is completed, the EA repository manager triggers an action to merge the import model of an information source to the EA model (UC6).	
Extensions	Step	Branching Action	
	1	The import is triggered by the federal system automatically based on a period specified in the configuration.	
	2	The EA repository manager wants to view differences of the import model compared to the EA model: View model differences (UC5).	

- information source events, e.g. number of updated records typically specified a priori, or
- external events, e.g. in the course of an acquisition the latest EA information is needed to merge application landscapes.

In their book, Bauer and Günzel [BG13, p.94] clarify that strictly speaking, periodic and request-based extractions can also be classified as event-based.

Instantaneous. To get the latest changes of an information source near real-time, an instantaneous propagation of modifications in an information source must be considered. Instantaneous synchronization demands high efforts to propagate changes to the federal system immediately as they occur in an information source.

Focusing on UC3, an import of information takes place in an automated manner. This keeps the import model up-to-date with reasonable efforts. In a federated EA model environment, the different synchronization strategies could be realized. The choice depends on the specific EA model and criticality of a particular model element with respect to the impact of model element changes on decision making.

UC3 further comprises a merge of the import model with the EA model. This use case is described separately (cf. UC6) in particular, since the EA repository manager may want to view the differences between an updated import model and an EA model prior to a merge, cf. UC3 and UC5.

4.2.5 Import Information

The transfer of information from an export model to an import model is subject to our next use case, UC4, described in Table 4.5. A particular challenge during the import of information is the detection of changes within an information source.

Especially in the context of DWH systems, the detection of *incremental changes* is regarded a necessity since the amount information that must be transferred to a DWH is too voluminous for a bulk load. Within the DWH community, Vavouras [VGD99], [Va02, pp.95–96], Vassiliadis [Va09], Kimball [KC04, p.106ff] as well as Bauer and Günzel [BG13, pp.54–55] report on the following possibilities to monitor an information source.

log-based assumes that the information source builds on a DBMS which captures transactions in a log file [Va02, pp.95–96]. This log file is analyzed using scraping or sniffing techniques [KC04, p.107]. Effectively, scraping takes the ‘*database redo log*’ and parses it for modifications on relevant information. Sniffing on the other hand polls on the redo log and captures modifications on the fly. Kimball and Caserta regard sniffing on log files as “probably the messiest of all techniques [in particular since they could] get full and prevent new transactions from occurring” [KC04, p.108].

trigger-based refers to an active mechanism and requires a particular trigger on a database event [Va02, p.95]. A particular suitable example is an update trigger, i.e. any modification of an information source could inform the federal system. Effectively, the

Table 4.5: Use Case: Import Information

Use case #	UC4	Use Case Name	Import Information
Context of use	An export model of an integrated information source should be synchronized with its import model within the federal system, e.g. during model synchronization (UC3).		
Scope	Federal system implementing MODELGLUE		
Level	Primary Task		
Primary actor	EA repository manager		
Stakeholder & Interests	Stakeholder	Interest	
	EA repository manager	wants to import changes a model of an information source to the federal system	
	Data owner	wants to share information for the common good of an enterprise	
	EA stakeholder	wants to have an up-to-date decision base	
Preconditions	The respective information source is integrated, i.e. a valid configuration file exists.		
Minimal Guarantees	The transaction is <i>'idempotent'</i> (see [Da04, p. 541]), i.e. another import, e.g. resuming an interrupted import, will only affect not yet updated model elements assuming the export model has not been changed. Any changes written to an import model are stored within a version history and can be reverted.		
Success Guarantees	All changes made to an export model within an information source since the last import are written to the respective import model within the federal system.		
Trigger	The EA repository manager starts this use case manually or the federal system triggers it automatically.		
Main Success Scenario	Step	Action	
	1	The EA repository manager chooses an action to start an import for a particular information source.	
	2	The federal system loads the configuration file for the chosen information source and connects to the specified destination.	
	3	The federal system reads changes of an export model according to queries defined in the configuration file.	
	4	The federal system executes the specified model-to-model transformation for each model element with respect to the mapping specification within the configuration file.	
	5	The federal system writes transformed model elements to the import model, i.e. new model elements are created.	
	6	For new model elements, the federal system creates a reference indicating the origin of the model element, i.e. a unique identifier within the information source.	

	Step	Branching Action
Extensions	1	The import is triggered by the federal system automatically.
	Step	Description
Technology & Data Variations	3	In the event the information source does not support to query for the last changes, information is read as a single bulk and differences to the previous version are calculated employing the import model within the federal system.
	5	The model element of the export model has a corresponding model element in the import model: The system detects existing model elements and updates information only in case the model element has been altered since the last import. This update is stored as a new version of the model element, i.e. old versions can still be retrieved.

propagation of changes is realized as a DBMS hook. Vavouras [Va02, p. 96] proposes to store the modification in an auxiliary table or file upon detection of modifications.

replication-based means one keeps an entire replica including all changes. This is similar to the proposal of Vavouras since operational information and changes thereof are stored separately. Vavouras states that commercial tools offer mechanisms to detect changes and offer replication services to store the information elsewhere [Va02, p. 96].

time-based can be regarded as a very intuitive way to query for changed data. Each piece of information is associated with a timestamp which is set to the current time in the event of a modification [BG13, p. 54]. Kimball and Caserta point out that although intuitive, the approach comes with a severe drawback. A time-based query loads duplicate information when it is restarted from a mid process failure. “This means that manual intervention and data cleanup is required if the process fails for any reason” [KC04, p. 108].

audit columns is a special variant of the time-based approach detailed by Kimball and Caserta [KC04, p. 106]. It utilizes so-called ‘*audit columns*’. These can be considered build-in attributes that track date and time as soon as a particular model element is created or modified. Kimball and Caserta report that information systems often already use audit columns. Thus, using audit columns would not necessarily modify the information source. Kimball and Caserta also warn about a particular threat to this method, namely back-end scripts that could modify these audit columns such that one could miss changes. As a countermeasure they propose is a quality assurance step for back-end scripts to ensure consistency of the audit columns.

snapshot-based involves a periodical data dump and a subsequent delta calculation, i.e. differencing of consecutive dumps, to capture changes [Va02, p. 96]. Depending on the polling frequency, a considerable amount of changes may remain undetected. Kimball and Caserta coin the term ‘*process of elimination*’ which describes the differencing of the previous version with the current data dump. The authors point

out that this is not the most efficient technique but “the most reliable of all incremental load techniques for capturing changed data” [KC04, p. 108].

application-assisted refers to the explicit adaptation of the underlying application logic of an information source. Vavouras argues that this option particularly must be considered for non-DBMSs and often is the only option for an incremental update of a DWH. A naïve approach is to alter the application to add timestamps in the event information is changed. Subsequent polls can then query for information that is annotated with a timestamp older than the one of the previous query. Vavouras further describes the creation of a ‘*delta file*’ that captures changes in an explicit manner.

Although the efforts are high, in line with Vavouras [Va02, p. 97], we advocate that only snapshot-based monitoring is applicable for most legacy systems and agree with Kimball and Caserta [KC04, p. 108] that snapshot-based monitoring is a very reliable technique.

As of now, we only considered to keep a copy, namely the import model stored within the federal system, that embraces the desired part of an information source’s model. A different approach is to establish semantic links to the information source (cf. [CHL⁺13]). Bergamaschi et al. [BCV99] advocate to establish (semantic) links between objects rather than transferring a deep copy of the information from one model to another. We briefly discuss the differences between maintaining links to objects in a model or to create copies of objects. Table 4.6 gives an overview of both approaches.

	Copy	Semantic Link
Versions/History	federal system	depends on the information source
Redundancy	redundant information	none, just identifiers
Differencing and conflict detection	queries information from the federal system	queries information from the information source
Synchronization efforts	high: queries (changed) information as a deep copy from the information source	low: queries for the links within the information source

Table 4.6: Semantic links vs. information import

In contrast to Bergamaschi et al. [BCV99], we advocate to keep a local copy within the federal system. This guarantees a version history and better autonomy of the import model and the information source’s model. Any additional modifications, aggregations, or cleansing can take place outside the information source and respective transformations do not influence the operative system. Additional features offered by (semantically) linked data, e.g. support for reasoning, is currently neither required nor are operationalized ontologies like RDF frequently found in legacy systems. Hence, UC4 (cf. Table 4.5) describes the import of information from an information source to the federal system rather than creating links to objects, which remain only within their original information source.

4.2.6 View Differences

Table 4.7: Use Case: View Differences

Use case #	UC5	Use Case Name	View Differences
Context of use	Prior merging models or for planning purposes, it may become necessary to view the differences between two models.		
Scope	Federal system implementing MODELGLUE		
Level	Primary Task		
Primary actor	EA repository manager		
Stakeholder & Interests	Stakeholder	Interest	
	EA repository manager	Wants to view the difference between two models prior merging or for planning purposes	
Preconditions	A model of an information source has been imported and at least two models exist within the federal system.		
Minimal Guarantees	Information is not altered by any user interaction.		
Success Guarantees	The differences between two models are visualized.		
Trigger	The EA repository manager starts this use case manually.		
Main Success Scenario	Step	Action	
	1	The EA repository manager selects two models within the federal system; commonly the EA model and an import model for an information source.	
	2	The system calculates the differences between the models.	
	3	The differences between the metamodels of the chosen models are shown to the user in a visual manner.	
	4	The user can drill down on model differences by clicking on a concept of the metamodel.	
	5	The system shows all instances of a chosen concept, i.e. instances of both models as well as differences between them.	
	6	The user can click on an instance to see its relationships to other instances.	
	7	The user can click on an instance that has been changed, i.e. has differences, in one or both models.	

In many publications, e.g. [SMR12, RHM13b, RHZ⁺13, RZM14], we emphasize the importance of visual means for EA management. In [RM14], we claim that viewing model differences is an essential task for enterprise architects. In Federated EA Model Management, prior to a merge of an import model with the EA model or in the course of modeling planned states, it may become necessary to view differences of two models. This way, the EA repository manager gets an overview which information has been changed during the last synchronization. However, no common standard to view differences in models and in particular for EA models exists and we observe a particular challenge in the communication

of model differences. Table 4.7 illustrates the respective use case (UC5). It embraces the calculation and the visual display of differences as well as the provision of means necessary for an analysis thereof. Differences thereby include both, model as well as metamodel differences.

When it comes to model differencing, to some extent one can learn from source code merging. Code merging is commonly supported by tools applying textual merging [Me02]. As we outlined in Section 2.3.1, there exist two general approaches to calculate differences, two-way differencing and three-way differencing. We detail our approach to model differencing in Section 5.2.4.

4.2.7 Merge Models

A unidirectional model synchronization can be viewed as a merge of models. Although UC6 (see Table 4.8) describes this use case in a generic way, we argue that in a federated EA model environment commonly one or more *import models* must be merged with an EA model, cf. [KR14].

Since we assume all modeling communities stay autonomously, the EA model as well as the model of the information source co-evolve. In Section 4.1.3, we outlined that a federated EA model environment contains boundary model elements (cf. Definition 4.7 on p. 83). “Conflicting representations are a major challenge for integration methodologies. Two designers modeling the same universe of discourse, or two overlapping universes of discourse, will probably describe the common real-world objects in different ways” [SPD92]. In the event of a merge of models within a federated EA model environment, conflicts may arise particularly for modifications on boundary model elements. Taentzer et al. outline that during a merge of different models incorporating concurrent changes conflicts may arise [TEL⁺10]. Wieland et al. [WLS⁺12] discuss that most of these conflicts cannot be resolved automatically. The modelers’ intentions, especially the semantics are not captured in the model.

In this vein, the federal system must store context information about a conflict. To resolve such model conflicts, the federal system sends *human tasks* containing a conflict description to roles, i.e. persons who are responsible for a particular model element. Conrad [Co97, p. 79] distinguishes between four kinds of conflicts originally published by Spaccapietra et al. [SPD92]. These are: semantic conflicts, descriptive conflict, heterogeneity conflicts, and structural conflicts.

semantic conflicts may arise if modelers describe common real-world object in different ways. Spaccapietra et al. [SPD92] name, for instance, that one view may denote a class STUDENT that matches the semantics of another view’s COMPUTER SCIENCE STUDENT. These semantic conflicts have been discussed by Mannino and Effelsberg [ME84] for the domain of relational models extensively. In a federated EA model environment, semantic conflicts are grown evolutionary. We regard the resolution of semantic conflicts as a highly challenging, manual, and collaborative task.

descriptive conflicts may arise if modelers describe a common real-world with different attributes due to a different perception or intended focuses. The information for

Table 4.8: Use Case: Merge Models

Use case #	UC6	Use Case Name	Merge Models
Context of use	After information has been imported to an import model in the federal system it may be synchronized with the EA model.		
Scope	Federal system implementing MODELGLUE; EA repository		
Level	Primary Task		
Primary actor	Enterprise architect (EA coordinator or EA repository manager)		
Stakeholder & Interests	Stakeholder	Interest	
	Enterprise architect	wants to update the EA model with information of an import model	
	EA stakeholder	wants to have a sound and up-to-date decision base	
Preconditions	Information has been imported to the corresponding import model in the federal system.		
Minimal Guarantees	The source models and the target model remain unchanged until the merge transaction is completed. The source model does not cease to exist after the merge transaction.		
Success Guarantees	One or more chosen source models are merged with a target model and all conflicts have been either resolved or forwarded to a responsible role.		
Trigger	The enterprise architect manually triggers an action to merge models.		
Main Success Scenario	Step	Action	
	1	The enterprise architect chooses models to be merged (source models) and a destination (target model). The source models are commonly one or more import models whereas the target model commonly is the EA model.	
	2	The federal system creates a preview model (a clone of the target model) that is used during the merge process such that subsequent actions have no immediate effect on involved models.	
	3	The federal system attempts to merge the source models with the target model.	
	4	The federal system detects conflicts that arise during a merge.	
	4a	The federal system resolves conflicts automatically if a respective rule is configured within the conflict resolution strategy.	
	4b	The federal system sends human tasks for any unresolved conflict for further analysis and resolution to responsible roles defined in the involved models.	
	5	The federal system checks for a corresponding model element within the source and preview model; if no conflict occur, it applies changes performed on the source model to the preview model.	
6	Enterprise architects and EA stakeholders resolve model conflicts (see UC7 and UC8).		

4. Requirements Analysis

Main Success Scenario (cont'd)	6a	On resolution, the federal system marks human tasks for conflicts as resolved.
	7	The enterprise architect finalizes the merge transaction.
	8	The federal system moves the preview model to the target of the merge action (commonly considered the EA model or a planned state thereof, both stored within the EA repository)
	9	For unresolved conflicts in the target model, human tasks remain in the federal system for further analysis and resolution by respective responsible roles.
Extensions	Step	Branching Action
	7	The enterprise architect aborts the merge process: the federal system deletes the preview model including human tasks that describe a conflict therein.
Technology & Data Variations	Step	Description
	8	The target model is stored within the federal system: the federal system moves the preview model to the target model.

a common real-world object in different domains and the information captured by the resulting models can differ considerably. Descriptive conflicts also include name conflicts, i.e. modelers using different names for the same semantics (synonyms) or using the same name for describing different semantics (homonyms).

heterogeneity conflicts may arise if the different information sources follow different paradigms to describe and store information (cf. Section 4.1.3). Although two modelers using a structured way to model real-world objects, they could use different approaches to model real-world objects, e.g. relational or object-oriented models. An ontology model offers more mechanisms, i.e. is semantically richer, than an object-oriented model, which in turn is semantically richer than a relational model. In a federated EA model environment, this kind of heterogeneity is ‘built-in’; it is a challenge enterprises face that comes with a legacy system environment.

structural conflicts may arise even if the same paradigm is used for modeling. Modelers can choose from a variety of alternatives to represent common real-world objects. For instance, in an object-oriented model, the same set of real-world objects may be represented as an entity type in one view and as an attribute of an entity type in another view.

For relational databases in multidatabase systems (MDBS), Kim and Seo [KS91] classify conflicts into schema conflicts and data conflicts. However, we foresee that in the course of an evolutionary EA model, even more complex conflicts will arise. That is in the terminology of the database community: conflicts between data, between schemata, and between schema and data. With respect to object-oriented information systems, we distinguish different categories of conflicts that may arise between models (see Definition 4.13).

DEFINITION 4.13: Model conflict

A conflict between two or more models may arise in the course of concurrent modeling. That is

- model/model conflicts, i.e. conflicts between two or more instances,
- model/metamodel conflicts, i.e. conflicts between one or more instances and the respective metamodel, and
- metamodel/metamodel, i.e. conflicts between two or more metamodels.



4.2.8 Resolve Conflict

In the course of synchronizing information sources, the EA repository manager may seize the chance to immediately resolve conflicts. Major reason to resolve conflicts is consistency between the different interacting models, cf. Assumption 4.7.

The resolution of these model conflicts is subject of our next use case illustrated in Table 4.9 on p. 106. In essence, it centers around the resolution of conflicts that arose during a model merge. Commonly, such a merge of models may include two models. Potentially, conflicts in more than two models could arise during an n-way merge of models [KR14].

In UC6, the federal system detects the conflicts and generates human tasks for unresolved conflicts. Each of these task contains a description of the conflict. This description is interpretable by a human actor and must contain sufficient contextual information to resolve the conflict. The user can either resolve the conflict or forward the task to another user that may be capable to resolve the conflict adequately.

Central to the use case is the notion of a *worklist*. Each user within the federal system has such a worklist. Essentially, it embraces a list of human tasks assigned to that specific user. A human task or *task* describes a piece of work that is assigned to one or more persons. The notion of tasks is further detailed in UC10. For now, we continue the discussion with an implicit understanding the term.

Two types of collaboration are addressed in UC7, i.e. asynchronous and synchronous. The former type is realized by forwarding a task which essentially transfers a task from one person's worklist to another person or assigns additional persons to a task.

Table 4.9: Use Case: Resolve Conflicts

Use Case #	UC7	Use Case Name	Resolve Conflicts
Context of Use	After information has been imported to a model and merged to an EA model conflicts may arise.		
Scope	Federal system implementing MODELGLUE		
Level	Primary Task		
Primary Actor	Enterprise architect		
Stakeholder & Interests	Stakeholder	Interest	
	Enterprise architect	wants to keep the EA model consistent such that it represents a reliable decision base within the organization	
	Data owner	wants to know whether information is conflicting and perhaps must be corrected in the information source	
EA stakeholder	wants to have a sound decision base with consistent information		
Preconditions	Information has been imported to a model and merged to an EA model such that conflicts arose.		
Minimal Guarantees	Any unresolved conflicts are assigned to at least one person that is responsible for its resolution.		
Success Guarantees	All conflicts have been either resolved or forwarded to a responsible person.		
Trigger	This use case is triggered by its stakeholders in the event of merging two or more models.		
Main Success Scenario	Step	Action	
	1	The user can choose a task that describes a conflict to be resolved from the worklist.	
	2	The user can view conflicting changes; for each change, the involved role or information system of an information source and temporal information of a change is given.	
	3	The user can apply a change as a resolution to the conflict or revert changes to the original state.	
	4	The federal system propagates the chosen change to the model.	
5	The task is marked as 'resolved' which denotes that the conflict is resolved; the task disappears in the worklist of each assigned role.		
Extensions	Step	Branching Action	
	3	The user does not know the final resolution to a conflict: the user annotates the conflict and forwards it to another user.	
	3	The user initiates a collaborative resolution session: resolve conflict collaboratively (UC8).	
3	The user wants to reschedule the conflict for further analysis: the user postpones the conflict, i.e. the task remains in the worklist.		

	Step	Description
Technology & Data Variations	1a	The user can choose from a list of conflicts by browsing the model.
	1b	The user can choose conflicts in a conflict management dashboard that displays conflicts within the model at the place at which they occur.
	4a	The conflict describes a conflict within an information source: the user resolves the conflict within the information source utilizing the contextual information. Subsequently the user alters the state of the task manually to indicate the resolution status of the conflict.

4.2.9 Resolve Conflicts Collaboratively

As outlined above, the resolution of conflicts between an information source and an EA model is a highly collaborative task. The next use case (UC8) detailed in Table 4.10 puts a special focus on synchronous collaboration in the course of model conflict resolution.

In UC3, we describe that the EA repository manager performs the synchronization triggered either manually or the synchronization is triggered automatically by the federal system, e.g. on a temporal event such as once a month, a week before the due date of the financial statement, etc. As an important step, the merge of models (UC6) is intended to bring together current models of potentially different information sources and the EA model. Conflicts may be resolved by a single user or by multiple users in a collaborative and asynchronous way (UC7).

In line with Wieland et al. [WLS⁺12], we advocate that collaboration is essential for the resolution of conflicts. A synchronous discussion can be regarded a highly efficient and effective means to resolve model conflicts collaborative. Especially effectiveness is a desirable property of conflict resolution within Federated EA Model Management and contributes to the goal to achieve CCMC. Thus, UC8 describes how the conflict resolution can take place as a collaborative session. This session may take place between the enterprise architects, data owner, and other EA stakeholders that may have an interest in resolving particular conflicts.

Key to collaborative conflict resolution is the notion of a *conflict management dashboard*. This dashboard shows information that is relevant to resolve the conflict directly in the modeling context. That embraces the exact location of a conflict within a model, the changes on a model that must be discussed, involved persons that issued these changes, and additional information about the modeling context. Basic chat facilities (messenger and voice) can support the synchronous resolution of model conflicts. They enable ad-hoc communication and provide an efficient means to collaborate while the model and information about the conflicts to be solved are always at hand and media-breaks can be avoided.

Table 4.10: Use Case: Resolve Conflicts Collaboratively

Use case #	UC8	Use Case Name	Resolve Conflicts Collaboratively
Context of use	After the merge of models conflicts may arise that can be solved best in a collaborative fashion.		
Scope	Federal system implementing MODELGLUE		
Level	Primary Task		
Primary actor	Enterprise architect		
Stakeholder & Interests	Stakeholder	Interest	
	Enterprise architect	wants to keep the EA model consistent such that it represents a reliable decision base within the organization	
	Data owner	wants to know whether information is conflicting and perhaps must be corrected in the information source in order to improve the reliability of the information source	
	EA stakeholder	wants to have a sound decision base with consistent information	
Preconditions	Information was imported to a model and merged to an EA model such that conflicts arose.		
Minimal Guarantees	Unresolved conflicts are assigned to the role that opened them.		
Success Guarantees	All conflicts are either resolved or forwarded to a responsible person.		
Trigger	Triggered manually by stakeholders after two or more models were merged.		
Main Success Scenario	Step	Action	
	1	The EA coordinator invites users (EA stakeholders and enterprise architects) to a collaborative session.	
	2	The users open the conflict management dashboard and join a collaborative session.	
	3	The federal system displays the conflict management dashboard that shows conflicts which have been detected and according to the conflict resolution strategy configured are not resolvable automatically.	
	4	Users can only view conflicts assigned to them either individually or due to membership of a group, i.e. everyone within the collaborative session on the one hand sees a personalized view but on the other sees interactions that are applicable for this personalized view.	
	5	Users can choose a conflict to be resolved.	
	6	Users can view conflicting changes, involved roles, and timestamps to discuss the conflict.	
7	The federal system synchronizes the views and actions of each user in real-time with respect to individual access rights.		

Main Success Scenario (cont'd)	8	Users can work on a conflict collaboratively.
	8a	Users can chat with each other to discuss a conflict.
	8b	Users can have a voice session with each other to discuss a conflict.
	8c	Users can annotate conflicts.
	8d	Users can forward conflicts.
	8e	Users can apply changes as a resolution to the conflict.
	8f	Users can discard changes as a resolution to a conflict.
	9	The federal system applies changes.
	9a	The federal system propagates chosen conflict resolutions to the model immediately.
	9b	The federal system persists open conflict for further analysis.
	10	The model conflict is marked as resolved and disappears in the model conflict list, the conflict management dashboard, and worklist of each assigned role.
Technology & Data Variations	Step	Description
	9a	A conflict resolution requires changes that affect an information source, i.e. changes influence the model or meta-model of an information source: The federal system generates 'propagate' tasks and sends it to the responsible role such that changes can be aligned.

4.2.10 Configure Conflict Resolution

Once decided on a conflict resolution, the participants of a collaborative conflict resolution session (UC8) may identify a certain pattern that could be applied in the future to resolve conflicts automatically. Such a pattern could be part of an organization-specific conflict resolution strategy. The configuration of predefined conflict resolution strategies and organization-specific adaptations are subject to our next use case, UC9. It is detailed in Table 4.11 on p. 110.

Wieland et al. [WLS⁺12] report on model conflicts that arise in the course of a merge of concurrently edited models with a common origin. The authors describe the problem when operations are simultaneously applied to model elements in different branches describing the same real-world objects. Because logically simultaneous operations overlap, the result might be a conflict. Table 4.12 on p. 111 summarizes central findings of Wieland et al. [WLS⁺12]. The table depicts overlapping operations and designates situations in which a conflict arises in the course of concurrent modeling activities and situations in which the modelers did not see all information and implicit changes took place. For some of these conflict situations, an organization-specific action can be taken to resolve the conflict automatically. However, since we distinguish three kinds of model conflict classes, i.e. model/model, model/metamodel, and metamodel/metamodel conflicts, conflict detection within a federated EA model environment tends to be more complex. Moreover, model conflicts in EA management are commonly resolved collaboratively and demand prior communication and possibly mediation. Thus, we conclude that while some model conflicts can be resolved specifying rules (cf. [KR14]),

Table 4.11: Use Case: Configure conflict resolution strategy

Use case #	UC9	Use Case Name	Configure conflict resolution strategy
Context of use	During conflict resolution (UC7 and UC8), the participants may identify a pattern that can be applied to resolve certain conflicts in the event of prospective model synchronizations.		
Scope	Federal system implementing MODELGLUE		
Level	Primary Task		
Primary actor	EA repository manager		
Stakeholder & Interests	Stakeholder	Interest	
	Data owner	wants to minimize efforts to keep the EA model consistent	
	EA repository manager	wants to specify or refine a rule within the current conflict resolution strategy	
	EA stakeholder	wants to have a sound decision base	
Preconditions	The federal system must be in operation and a conflict pattern must have been identified.		
Minimal Guarantees	A syntactically valid conflict resolution strategy is stored within the federal system.		
Success Guarantees	The new conflict resolution strategy reflects user needs, is persisted, and is applied on subsequent model merges.		
Trigger	The EA repository manager triggers this use case manually in the event the conflict resolution strategy must be altered.		
Main Success Scenario	Step	Action	
	1	The EA repository manager opens the customization dialog for the current conflict resolution strategy.	
	2	The EA repository manager can choose from predefined strategies: A strict strategy with notifications for deletions and concurrent updates, or an optimistic and more conflict tolerant strategy with less notifications on deletions and concurrent updates. Both strategies realize a lossless conflict resolution.	
	3	The federal system displays the behavior for each model event including potentially conflicting events for the chosen strategy to the user.	
	4	The EA repository manager can choose from model events to customize the strategy currently active.	
	5	The federal system displays a customization dialog that facilitates the configuration of organization-specific merge rules.	
	6	The EA repository manager specifies merge rules that are applied on a specific model event. Commonly this is done only for conflicting situations, e.g. for simultaneous delete/update operations on the same model element.	

Main Success Scenario (cont'd)	7	The federal system persists the altered conflict resolution strategy with the merge rules specified.
Extensions	Step	Branching Action
	2	The EA repository manager chooses to modify the existing strategy: The federal system loads the existing conflict resolution strategy.

most model conflicts within EA management must be resolved manually, i.e. they require human intervention.

	Insert	Delete	Update	Use	Move
Insert					
Delete			×	×	×
Update			×	!	!
Use					!
Move					×

×: Conflict
!: Warning

Table 4.12: Overlapping changes according to Wieland et al. [WLS⁺12]

Common resolution patterns in this context include but are not limited to:

- take change of a particular information source over another,
- take latest change,
- take changes submitted by person *A* over person *B*, or
- merge both changes violating constraints.

In particular the last pattern can be incorporated in a predefined, *tolerant merge strategy*. Goal of such a tolerant merge strategy is to store any information in a human interpretable manner while eventually violating *federal consistency*.

In contrast to such an tolerant merge strategy, a *strict merge strategy* does not violate federal consistency and immediately generates conflict tasks as modelers might not have seen all information that has been modeled concurrently. That includes many cases in which persons have to be informed about a change or must confirm a deletion. In UC9, one of two predefined conflict resolution strategies, i.e. tolerant and strict merging, can be modified by an expert such that an organization-specific merge strategy evolves over time. Additionally, this organization-specific strategy can be loaded and tailored iteratively. Such an organization-specific conflict resolution strategy may cope with conflicts in an increasingly automated manner or account for organization-specific exceptions in models or metamodels.

**EXAMPLE 4.6: Organization-Specific Conflict Resolution**

The Database Administrator (DBA), John Doe, recognizes that information about some physical servers stored within the CMDB is flawed. John informs Sally, the data owner of the CMDB. Sally recognizes a systematic error within the CMDB that can be traced to a particular revision number. It seems that the procurement department did not enter the right revision numbers after the acquisition of these machines. However, Sally sees that the CMDB is valid for revision numbers after 0.2. The CMDB is highly integrated and, thus, Sally decides for local inconsistency, i.e. to maintain this inconsistent state. As a data owner of an integrated information source, Sally knows that information contained in the CMDB is used for decision making by EA management. Hence, she writes a brief e-mail to Bob, the EA repository manager. The e-mail contains information about how to deal with these flaws in information. Receiving the e-mail, Bob immediately knows how to resolve this issue for the EA model; he would just use another CMDB as primary information source for these particular servers and merge the two CMDB models with the EA model. He takes a brief note that sketches the rule for the subsequent conflict resolution.

```
1 if(A==B && CMDB1.server.revision <= 0.2)
2   revision = CMDB2.server.revision
```



4.2.11 Assign Task

The next use case details the assignment of tasks to individuals or groups. In Federated EA Model Management this assignment can take place either manually or automatically. For the manual assignment, we assume that the EA repository manager could detect model inconsistencies in the model at any time (Section 5.2.6). A particular example for such a situation is the detection of an *abstraction gap*. In [RHM13b] we discuss the role of an abstraction gap; in the present thesis we postpone the discussion of this phenomenon and detail its specifics in Section 5.2.8.4. Other triggering events that could lead to an assignment of a task within Federated EA Model Management center around the resolution of model conflicts. An assignment or reassignment (forwarding) of tasks could take place in the event of resolving conflicts (UC7) or during a collaborative conflict resolution session (UC8). An automated assignment of tasks on the other hand is initiated by the federal system. This may happen during the merge of models if the federal system detects conflicts (UC6). In [RHM13a], we detail other events that may trigger an automated generation of tasks in the course of EA model and metamodel evolution.

In line with Malone and Crowston [MC94], we agree that a task means both, achieving goals and performing activities (cf. also [MCH03, ch. 3.2.3]). Goals describe desired states of the real-world whereas activities refer to actions performed to achieve a particular state. Malone et al. advocate that both, goals and activities, are clearly different. However, the authors further argue that analyzing both concepts together makes sense. They compose goals into

Table 4.13: Use Case: Assign Task

Use case #	UC10	Use Case Name	Assign Task
Context of use	A model conflict has been detected or annotated.		
Scope	Federal system implementing MODELGLUE		
Level	Primary Task		
Primary actor	EA repository manager		
Stakeholder & Interests	Stakeholder	Interest	
	Data owner	wants to have the right information in the information source such that no further conflicts occur and information does not contradict other statements about the real-world	
	EA coordinator	wants to delegate existing conflict tasks to resolve conflicts	
	EA repository manager	wants to enforce consistency within the federation by delegating tasks that describe activities to resolve issues, i.e. inconsistencies, in a model	
	EA stakeholder	wants to have a sound decision base with consistent information	
	EA modeling expert	wants to facilitate the resolution of conflicts found manually	
Preconditions	A model conflict has been detected.		
Minimal Guarantees	The federal system does not allow to persist any unassigned tasks.		
Success Guarantees	An existing task or a new task is (re)assigned to a user.		
Trigger	The EA repository manager or the federal system detects a model conflict.		
Main Success Scenario	Step	Action	
	1	The EA repository manager or the EA modeling expert chooses an action to create a new task.	
	2	The federal system creates a new task that is in an initial state.	
	3	The EA repository manager adds a description to the task that explains the issue to be solved and its context.	
	4	The EA repository manager assigns the newly created task to an individual or group that is responsible for the issue to be solved.	
	5	The task is in the state assigned and is shown in the assignee's worklist.	
Extensions	Step	Branching Action	
	1	The EA repository manager or another enterprise architect chooses an existing task that has to be reassigned.	
	1	The federal system creates the task during a merge of models (UC6).	

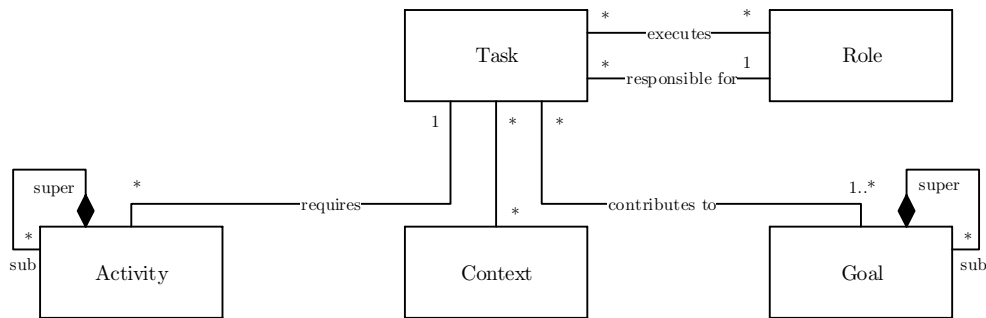


Figure 4.11: An initial conceptual notion of tasks within Federated EA Model Management

subgoals to be achieved and activities into primitive sub-activities to be performed. This way, Malone et al. describe the notion of a task that embraces both, goals and activities to be undertaken.

Figure 4.11 illustrates a conceptualization of this view. Additionally, we add the notion of **CONTEXT** in which **ACTIVITIES** are carried out and **GOALS** are achieved. Moreover, someone is always responsible that an activity is carried out and goals are achieved within a given **CONTEXT**, i.e. a state of the real-world. This **ROLE** not necessarily is the one that actually carries out an **ACTIVITY**, since **TASKS** can be forwarded, i.e. delegated to other **ROLES**. However, a certain understanding of the activity comes with being responsible for it. We conclude the discussion about tasks with Assumption 4.8.

ASSUMPTION 4.8: Responsible roles

We assume that responsible roles know how a task is performed and to whom it can be delegated.

4.2.12 View Conflicts

Contextual information of a task embraces any information about conflicts that facilitates their resolution. To resolve a conflict one must first view all relevant information about a conflict. This use case is further detailed in UC11, cf. Table 4.14.

The user goal of this use case is to get an overview of all conflicts as well as detailed information of a single conflict of interest. That includes information on

- **what** has been modified, i.e. which changes on which model element have been performed?
- **who** modified a model element, i.e. which user including contact details within the federal system?
- **when** did the modifications take place, i.e. what is the issuing date and time of the conflicting changes?

Table 4.14: Use Case: View Conflicts

Use case #	UC11	Use Case Name	View Conflicts
Context of use	A model conflict was detected and its resolution was assigned as a tasks to a responsible role.		
Scope	Federal system implementing MODELGLUE		
Level	Primary Task		
Primary actor	EA repository manager		
Stakeholder & Interests	Stakeholder	Interest	
	EA repository manager	wants enforce consistency within the federation	
	Data owner	wants view and eliminate the root cause of a conflict in the information source	
	EA stakeholder	wants to have a sound decision base with consistent information	
Preconditions	The resolution of a conflict task addresses a model element stored either externally, i.e. within an information source, or within internally, i.e. within the federal system.		
Minimal Guarantees	The federal system keeps the current information.		
Success Guarantees	Information about conflicts such as the origin of the affected model element and issued changes is shown to the user.		
Trigger	Stakeholders start this use case manually, preferably when they are about to solve a conflict (UC7) or delegate a task (UC10).		
Main Success Scenario	Step	Action	
	1	The user chooses an action to view all conflicts.	
	2	The system shows a list of conflicts.	
	3	The user can drill down on a single conflict to get its entire details.	
Technology & Data Variations	Step	Description	
	2	The system shows the conflicts as annotations on an UML-like diagram.	

- **where** could the modification exert influence on other model elements, i.e. what is the model element's context and does it have an impact on other model elements?

When an enterprise architect or an EA stakeholder is in possession of this information, they can decide how to cope with a model conflict (cf. UC7, UC8, and UC9); alternatively, they can forward the conflict to a person who may find an adequate resolution for this conflict (cf. UC10).

4.2.13 Additional Use Cases within Federated EA Model Management

Besides use cases that concern the federal system primarily, there are other use cases that center around the EA repository and the integrated information sources. Since the present thesis focuses on Federated EA Model Management and does not outline or detail a reference architecture for an information source or an EA repository, in this section we briefly sketch these use cases highlighting how they interact with UC1–UC11.

Note that stakeholders of an information source may raise additional use cases; the following use cases must be considered a non-exhaustive list that serves the further discourse of this thesis. We refer the interested reader to Matthes et al. [MBL⁺08]; they provide a comprehensive list and detailed descriptions of typical scenarios that are carried out using an EA repository. In Federated EA Model Management, the following use cases of an information source are essential to be addressed.

- **Manage information** embraces use cases that center around general knowledge management employing an information system. That includes the maintenance of a model that reflects the real-world. This model conforms to a metamodel that may exist implicitly or is stored explicitly within the information source.
 - **Keep model consistent** addresses the need to keep stored information in accordance with the real-world; the users of an information source naturally are focused to foster local consistency.
 - **Publish information** is an important use case, relevant for instance in the course of executive presentations as well as to other presentations to third parties that are interested in information and may benefit from it. This especially includes EA management and respective EA stakeholders.
- **Manage meta-information** covers all use cases that concern meta-information from managing access rights to tagging information. The former may apply to the model as well as metamodel whereas the latter commonly applies to the model only.
 - **Restrict information to be published** is especially important for Federated EA Model Management. Data owners may not want to publish every bit of information and want to restrict access to some concepts. This restriction can be either implemented by the information source or is part of the mapping (UC2).
- **Evolve metamodel** accounts for change requests on an information system that may alter the metamodel of an information source to adapt to an ever changing

environment. While this is not a day-to-day use case, we consider this an important use case as it has an influence on a coherent Federated EA Model Management. The metamodel evolution of an information source may require to adapt the export model, the import model as well as respective mappings (cf. UC1 and UC2).

- **Resolve conflict** in an information source is a use case specifically introduced to achieve CCMC within a federated EA model environment. A person may get a task assigned (UC10) that describes a conflict, which must be resolved within an information source. This resolution can be carried out by the common operations to alter information within an information source (manage information) such that the information system does not have to be changed to enter a federation.

An important term in these use cases is *meta-information*. Kimball defines meta-information as “all the information [...] that is not the actual data itself” [KR02, pp. 14–15]. In line with the ISO standard 11179 [ISO04, p. 10] we advocate that meta-information depends on particular circumstances, for particular purposes, and with certain perspectives. Therefore, Definition 4.14 seeks to give a more precise description of the term in the context of Federated EA Model Management.

DEFINITION 4.14: Meta-information[†]

The term meta-information refers to information that intends to describe information or properties thereof. For instance its

- creation and origin,
- flow and transition,
- ownership and access rights,
- ...

[†]Synonym(s): metadata [Ba68, p. 28], [KR02, pp. 14–15], [Na04], [ISO04, p. 10] ■

Commonly, meta-information cannot be derived from stored information and, thus, is maintained explicitly. Example 4.7 gives the reader an illustrative case in which we distinguish between information and meta-information.



EXAMPLE 4.7: information vs. meta-information

On Monday July 9, Bob calls Alice and tells her that he will arrive a bit later to an appointment. The information in this call is that Bob will arrive later, while the meta-information is any information about the fact that Bob called Alice using a particular device and phone number on Monday morning, July 9, 3.00 am.



Our final considerations for use cases within a federated EA model environment center around the EA repository. Let us assume that an EA repository can be viewed as a special kind of information source. Our recent study shows that more and more tools EA repositories store the metamodel explicitly [RZM14, p.22] and also allow user adaptations [RZM14, p.22]. Common use cases relevant to Federated EA Model Management are managing information and evolving the metamodel of an EA model. The former use case embraces the basic maintenance of an EA model. The latter accounts for the iterative nature of EA management and the evolution of an EA metamodel (cf. Section 2.1). Besides enterprise architects, EA stakeholders may have an interest to view information from time to time. This holds especially true in the course of conflict resolution. When EA stakeholders are about to contribute to a resolution of a model conflict, it is very likely that they must view information that is related to the conflict.

4.3 Requirements for a Federated EA Model Management Solution

Above, we presented use cases that build the foundation for a more detailed requirements analysis for a federal system. Use cases can be considered black-box behavioral requirements [Co01, ch. 17.3]. In [Co01, ch. 16], Cockburn states use cases are not sufficient and further information on requirements is needed to design and develop a solution. Hence, in this section, we deduce concrete requirements from these use cases introduced in Section 4.2 with respect to the challenges, we identified in an empirical study published in [HMR12]. Thereby, we employ some key words to indicate the requirements levels as defined by Bradner in Requests for Comments (RFC) 2119.

“The key words ‘MUST’, ‘MUST NOT’, ‘REQUIRED’, ‘SHALL’, ‘SHALL NOT’, ‘SHOULD’, ‘SHOULD NOT’, ‘RECOMMENDED’, ‘MAY’, and ‘OPTIONAL’ in this document are to be interpreted as described in RFC 2119.” [Br97]

The requirements are split into different color-coded categories, i.e. requirements focusing on

- Process requirements— are colored with a light-orange background,
- Collaboration requirements— are colored with a light-purple background,
- Modeling requirements— are colored with a light-green background,
- Usability requirements— are colored with a light-red background, and
- Technology requirements— are colored with a light-yellow background.

In the subsequent sections, we motivate each category briefly and discuss concrete requirements for each of these categories.

4.3.1 Process Requirements

In [RHM13b], we propose a collaborative process for the resolution of conflicts in EA models. Although this process has an iterative nature and incorporates escalation mechanisms, we regard this process as one possible success scenario for the resolution of conflicts. Deviations to this process are very likely and in the article we focus on the main success scenario to emphasize the collaborative nature of this process. Fahland and van der Aalst [FA13] diagnose that industrial business processes exhibit a plethora of exceptional behavior that commonly is not captured in a business process designed top down. Researchers in the domain of Adaptive Case Management (ACM), e.g. [Sw10, ch. 3], give experts—commonly knowledge workers—a high degree of freedom. These researchers discourage from anticipating each possible transition within a process. Instead ACM provides means such that each expert may treat a case differently and in turn each case may be treated differently by an expert. However, an information system can still provide utility for the management of each case, e.g. templates may facilitate tasks of knowledge workers and, thus, increase productivity [Sw10, ch. 7]. We advocate that conflict resolution in EA models cannot be supported by a ‘hardwired’ process that is executed by a common workflow or business process execution engine.

REQUIREMENT PR1: Non-deterministic process support

A solution must feature flexible process capabilities to facilitate an iterative and non-deterministic conflict resolution process with human tasks.

We consider it very unlikely that every conflict found either by a federal system, an enterprise architect, or by an EA stakeholder can be resolved immediately by a single person without prior consultation of other parties. Our next requirement focuses on the escalation in the event no resolution is found. Although the federal system must feature flexible process capabilities (cf. PR1), it should rely on a coherent role model that captures access rights and responsibilities such that a human task can be delegated along a chain of responsibility. Our next requirement focuses on the escalation in a chain of responsible roles in an enterprise (cf. [FAW07]).

REQUIREMENT PR2: Escalation mechanisms

A conflict resolution must define clear escalation mechanisms, i.e. a chain of responsibility for each conflict.

REQUIREMENT PR2 implies that the federal system is able to determine the responsible role of one model element on the finest possible granularity. We advocate this should be the case even if responsibilities are not set explicitly. This can be accomplished based on a chain of responsibility that divides work between different responsible roles. Gamma et al. describe the object behavioral pattern coined ‘*chain of responsibility*’ in [GHJ⁺94, p. 223ff]. Abstractly speaking, the pattern circumvents “coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it” [GHJ⁺94, p. 223]. Transferring this idea to a design for a federal system that facilitates Federated

EA Model Management, an object (person) handles a request (human task). To resolve a conflict, the path of this chain must be based on 1) the last-editors of a model element, 2) the explicitly named responsible persons, or 3) the data owner of a particular model element.

REQUIREMENT PR3: Clearly defined responsible roles on model elements

The system must be able to assign responsible roles on model elements at different granularities, i.e. elements of the model as well as elements of the metamodel.

A holistic solution design for Federated EA Model Management requires not only a federal system but also an adaptation of organizational processes [FAW07]. Towards a socio-technical solution for Federated EA Model Management, a description of required processes is needed. A solution should embrace an administrative process, involved roles, responsibilities, and triggering events. This process must give guidance throughout the integration of information sources in a federation and address the collaborative resolution of conflicts that arise during the synchronization of different models.

REQUIREMENT PR4: Process guidelines

A solution must describe the required core process steps to support and guide Federated EA Model Management.

4.3.2 Collaboration Requirements

As discussed before, an integration of a model in a federated EA model environment is a long-lasting process. In particular the conflict resolution (UC7 and UC8) is characterized by its collaborative nature with the ultimate goal to reach a consistent state within and among the models that participate in a federation, i.e. to achieve CCMC. In line with Bente et al. [BBL12, p. 137], we advocate that collaboration is essential for a successful EA management initiative [RHM13a]. Further, we agree with Wieland et al. [WLS⁺12] that it is equally important during the resolution of model conflicts [FAB⁺11b, RHM13b].

REQUIREMENT CO1: Collaborative conflict resolution

The system must provide facilities to resolve conflicts collaboratively by involving different stakeholders.

Although the resolution of model conflicts in Federated EA Model Management should take place collaboratively, it is essential to know which person is responsible to resolve a conflict ultimately. This is especially important during asynchronous collaboration and conflict resolution (cf. UC7). The next requirement addresses the determination of responsibilities for a model element which are utilized during the task assignment (cf. PR3).

REQUIREMENT CO2: Responsibility for conflicts

The system must be able to determine the responsible role for model conflicts in a chain of responsibilities.

EA management serves as a mediator to resolve conflicts utilizing tasks assigned to one role, which can be either a person or an entire group. A shared goal of the Federated EA Modeling Community, is to resolve conflicts (cf. UC7). In case this conflict resolution is done asynchronously an individual may not be able to resolve a conflict. However, a person could add new insights and relevant information (cf. CO3) that could contribute to the resolution of the conflict. In search of a resolution, both, synchronous and asynchronous collaboration should be used to discuss and resolve a conflict. Such a discussion can take place interactively in a synchronous manner or via annotations and comments asynchronously (cf. [Re14]).

REQUIREMENT CO3: Annotate and discuss conflicts

The system must provide means to annotate and discuss a conflict. A discussion may take place synchronously or asynchronously.

A person that is not in possession of all relevant information and cannot resolve a conflict should be able to delegate this conflict to another person possibly along the chain of responsibility (cf. PR3 and [FAW07]).

REQUIREMENT CO4: Delegation of conflicts

The system must be able to delegate contextual information and the responsibility for the resolution of conflicts to other roles.

Our next requirement seeks to resolve a conflict in a mediation session. Such a session typically takes place in a real-time collaboration tool, cf. e.g. [Ci14, Re14, Te14]. On the one hand this is similar to screen sharing, but on the other hand the federal system must respect the access rights of the different parties. That is, not every party should see all model elements and respective conflicts.

REQUIREMENT CO5: Real time collaboration

The system must provide facilities to discuss and resolve conflicts collaboratively. The discussion should take place by involving stakeholders in a real-time session. [FAB⁺11b], Own experience

In this vein, access rights and trust plays an important role. To a certain extent the need to share information is contradicting with the desire to restrict access to parts thereof. However, we regard access control as an essential property for a functioning Federated EA Model Management. This access control must be consistent in a shared session or deactivated explicitly.

REQUIREMENT CO6: Access rights during real time collaboration

The system must provide mechanisms to hold a synchronized collaborative session while enforcing individual access rights such that each participant is effectively limited to a personalized view, i.e. each participant must only see information according to respective access rights. [FAB⁺11b], Own experience

4.3.3 Modeling Requirements

As described in UC2, a mapping is created in a collaborative workshop. We assume that this mapping is created manually since we consider an application of semantic technologies to automate a mapping using a reference ontology, e.g. [NP01], not to be realistic considering the legacy systems that are involved in a federated EA model environment (cf. Assumption 4.2). However, this mapping must be translated into concrete model transformations.

REQUIREMENT MO1: Configurable model mappings

The system must provide means to define model mappings. Thereby, the translation of a logical mapping to physical mappings should be defined thoroughly.

In UC2, we describe that a logical as well as a physical mapping is specified. For a synchronization of models (UC3) the federal system must be able to interpret these physical mappings and act accordingly. The next requirement addresses the processing of the model mapping, i.e. the execution of model-to-model transformations.

REQUIREMENT MO2: Executable model mappings

The system must be able to execute model mappings, i.e. to translate model mappings into model transformations.

In UC1, UC3, and UC4 we assume that the federal system is able to create a model and metamodel for each information source. Moreover, in Section 4.2.13 we outline that an information source's metamodel may evolve over time and, thus, the mapping as well as the metamodel within the federal system has to be aligned.

REQUIREMENT MO3: Creation of a metamodel

The system must provide mechanisms to define a metamodel for each integrated information source.

In UC4, the users intent is to import information to the federal system. Regardless of the direction, i.e. push or pull, logically speaking the information must be imported into the federal system. Gathered information must be stored in a model within the federal system.

REQUIREMENT MO4: Import model

The system must be able to create and maintain an import model that conforms to a predefined metamodel.

In their best-selling book, Peters and Waterman [PW04, pp. 106] emphasize the importance of evolution and adaptation for superior organizations. This evolution has to be reflected by the EA and respective models describing the EA. We argue that a federal system must also be designed to maintain metamodels that evolve over time. In [BMM⁺11a, HRS⁺14, RHM13a], we advocate that an iterative approach to EA management prevails in industry. Especially an integration of multiple information sources is considered to last a longer period. Such an endeavor requires an incremental integration of information sources. Thus, the EA model and respective metamodel must be extended iteratively, i.e. they evolve. Additionally, in Assumption 4.6 we stated that the EA repository must be able to evolve its metamodel. Although Assumption 4.5 states models of information sources always conform to their metamodels, a solution must be designed such that similar to the EA repository, the information sources can evolve their metamodels. That implies an approach to Federated EA Model Management should not assume local consistency.

REQUIREMENT MO5: Evolutionary metamodel

The system must provide means to alter the metamodel without further migration steps.

To meet MO5, one can either strive for a coupled evolution, a so-called *co-evolution* of models and metamodels or through allowing and tolerating inconsistencies between models and their metamodels. Gruschko et al. [GKP07] present concepts for the coupled evolution of a model with a changing metamodel. In their article, they discuss the challenge of deducing automated migration steps of a model from metamodel changes. In the domain of Relational Database Management System (RDBMS), Terwilliger et al. [TBU10] study scenarios in which the metamodel and conforming models must co-evolve. They employ an explicit object-relational mapping between model and metamodel to support co-evolution and present a technique that, according to the authors in most cases, allows evolutions to progress automatically. In the present thesis, we take a different perspective. In contrast to the authors, we advocate a decoupled approach for Federated EA Model Management. On metamodel changes, human tasks should be utilized to inform relevant stakeholders and—if necessary—manual migration steps are employed by experts to carry out migrations (cf. [RHM13a]). Although an information source should maintain a model that is consistent with its metamodels, sometimes there could be inconsistencies within an information source. Davenport et al. [DHM10, p. 31] report that although inconsistencies within information pose a problem for analytics, skilled analysts can cope with inconsistent and flawed information. Thus, a solution should deal with these inconsistencies in a tolerant manner. That is, the information must be kept in any case and users decide how to cope with inconsistencies. However, these inconsistencies should only remain temporarily and resolved over time.

REQUIREMENT MO6: Offer the necessary degree of freedom

The system must allow maintaining an inconsistent model, i.e. a model that does not fully conform to its metamodel.

In UC5, stakeholders want to view the differences of a model. These differences have to be calculated in a process called differencing. This is subject to our next requirement.

Thereby, we consider that not only the model but also the metamodel changes over time potentially [RM14].

REQUIREMENT MO7: Model differencing

The system must be able to show the differences of two models with respect to their origin. This should not only include model differences but also metamodel differences.

Once, the differences are viewed and reviewed, the model of an information source can be merged with the EA model. Lippe and van Oosterom [LO92] as well as Koegel et al. [KHL⁺10] report that commonly operation-based approaches outperform state-based approaches. Operation-based approaches capture all model events whereas state-based approaches may miss modeling events or cannot distinguish between an update and a delete followed by a create operation.

REQUIREMENT MO8: Model merging

The system must be able to merge multiple (source) models to a given (target) model based on the operations, i.e. changes, performed on the source models.

Conflicts may arise during a model merge [KR14]. In particular concurrently describing the same real-world object in different models provokes conflicts.

REQUIREMENT MO9: Conflict detection

The system must detect conflicts that arise through concurrent modeling automatically. Further, the system should provide means to raise a model conflict manually for conflicts that are detected by a modeling experts.

The resolution of conflicts (cf. UC7 and UC8) is a long-lasting process [RHM13b]. Especially finding and agreeing upon patterns to resolve conflicts (cf. UC9) can be regarded as a non-trivial task that requires expert knowledge. However, an information system can still provide utility in such situations.

REQUIREMENT MO10: Conflict resolution

Conflicts must be delegated to responsible roles automatically or resolved by a user-defined conflict resolution strategy.

In Federated EA Model Management, conflicts between the EA model and other information sources should be resolved in the respective information source in which the information is stored. Thus, the federal system must provide a way to identify the respective origin of a model element [FAB⁺11b, FAW07]. This concerns the EA model as well as models of integrated information sources.

REQUIREMENT MO11: Identity reconciliation

The system must feature support for identity reconciliation.

As stated above, the necessary degree of freedom should allow users to maintain inconsistent information. However, over time, this situation should be improved. Next, we focus on the measurement of progress during conflict resolution and the degree of consistency of a model to its metamodel.

REQUIREMENT MO12: Model consistency check

The system should provide means to perform a consistency check of models. These checks embrace local as well as federal consistency verification.

In order to track changes and to guarantee traceability, the federal system has to keep versions of changes of different models that are integrated. This information can be utilized to get contextual information of a conflict, e.g. who or which information source changed what information at which particular date and time.

REQUIREMENT MO13: Version history

The system must be able to store versions of a model and should allow to restore a previous state of a model, i.e. revert changes.

Our next requirement addresses the access rights within the federal system. That includes the visibility of conflicts in a collaborative conflict resolution session (UC8), task assignment (UC10) and other use cases. Although these features are intended to create utility, they should not violate access rights. In [BMR⁺10b, BMM⁺11b], we discussed access right issues in the context of EA management. For a solution design of a federal system, we deliberate on the right granularity at which to define access rights. With regard to the collaborative nature of the conflict resolution, a coarse-grained access control may hide too much information. On the other hand, too fine-grained access control mechanisms may generate unnecessary administrative overhead. Shen and Dewan motivate such a fine-grained access control with respect to collaborative environments. They state the system “should allow independent specification of each access right of each user on each object” [SD92]. In line with the authors, we also recommend an easy assignment of access rights. Moreover, and in line with Saltzer and Schroeder, we advocate “fail-safe defaults” [Sa74, SS75].

REQUIREMENT MO14: Access rights

The system must be able to specify access rights for the model as well as metamodel on different levels of granularity. The assignment of access rights should incorporate an inheritance mechanisms and default access rights.

4.3.4 Usability Requirements

The human brain and human cognition is designed best for finding patterns in information presented visually [CMS99, Sp01, Tu01, Wa12]. This may be one reason why visualizations are a de facto means employed for communicating, past, current, planned, and target states of an EA. For this reason, the concepts of Federated EA Model Management should also focus on visual communication of model differences and conflicts.

REQUIREMENT US1: Understanding differences and conflict resolution
Users of the solution must understand mechanisms for displaying model differences and conflicts as well as the resolution thereof.

The users must have an intuitive understanding how to resolve conflicts. It is not only beneficial for an organization if a solution requires very limited training, but also is an advantage for users that utilize the system rarely, e.g. data owner and other EA stakeholders that are involved in the resolution process.

REQUIREMENT US2: Intuitive conflict resolution
Mechanisms for conflict resolution should be intuitive for the users.

In [Gr64, p. 857], Gross mentions the phenomena of ‘*information overload*’. Speier et al. give a sound definition of the term.

“Information overload occurs when the amount of input to a system exceeds its processing capacity. Decision makers have fairly limited cognitive processing capacity. Consequently, when information overload occurs, it is likely that a reduction in decision quality will occur.” [SVV99]

Yang et al. [YCH03] detail this problem in the context of information represented visually. The authors propose to employ fisheye views to cope with the problem of information overload. Kirschner [Ki12] seeks to cope with the problem for visualizations in the domain of EA management. He employs a helicopter view that seeks to give a holistic overview while allowing users to zoom into details. We conclude that mechanisms to deal with common model sizes in EA management must be provided such that the users do not lose overview.

REQUIREMENT US3: Coping with information overload
The system should provide means to cope with information overload.

At the same time, a certain degree of efficiency is also important. In particular communicating multiple conflict efficiently is an important aspect. This is subject to our next requirement.

REQUIREMENT US4: Batch processing—communication

The system should be able to facilitate the communication of multiple model conflicts.

Another important aspect intended to increase efficiency is to provide mechanisms that facilitate to resolve multiple conflicts at once, i.e. resolving multiple conflicts with just a few actions should also be possible.

REQUIREMENT US5: Batch processing—conflict resolution

The system should be able to define multiple conflict resolutions at once.

A particular means to facilitate the resolution of multiple conflicts in just a few steps is mechanism that learns from user input and recommends further actions based upon previous decisions, i.e. a heuristic. In combination with Us5, our next requirement seeks to eliminate unnecessary manual work.

REQUIREMENT US6: Learning

The system must be able to learn from previous decisions of modelers and propose recommendations for the resolution of conflicts.

Working on complex tasks is often designated by human errors, mistakes, and flaws. A typical example for such a complex task is the conflict resolution in the course of a model merge. If—for any reason—things go wrong, the user must be empowered to take counter actions with just a few actions. That is, a complete roll back of a merge action without any impact on the involved source and target models.

REQUIREMENT US7: Batch Processing—Correction

The system must support the user to take counter actions for multiple model conflict resolutions.

Although computer-mediated communication is often able to produce valuable results [Bo97], we regard face-to-face communication as highly effective means to develop models multiple parties agree upon.

REQUIREMENT US8: Face-to-face meeting support

The solution must be able to facilitate face-to-face communication.

4.3.5 Technology Requirements

The next category of requirements centers around technology constraints. During our work on the EA Visualization Tool Survey 2014 [RZM14], we perceived that there is a technology

shift from fat-client solutions towards web-based EA tools. Especially, this holds true for solutions recently developed.

In [Mc06], McAfee outlines the striving success of the web. Social platforms demonstrated that collaboration and knowledge exchange can be facilitated by web-based technologies. McAfee transfers the underlying technology, collectively summarized as Web 2.0 to collaboration requirements within an enterprise coining the term Enterprise 2.0. In line with McAfee, we regard Enterprise 2.0 tools to facilitate unstructured work that requires a high degree of collaboration and at the same time is knowledge intensive. McAfee as well as Davenport [Da05, p. 16] advocate the strong demand of autonomy when carrying out such work. A platform that has a low entrance barrier as well as an intuitive design (cf. Us2–Us8) can provide utility for organizations pursuing Federated EA Model Management. Such a technical solution also comes with a certain set of technology involved which is subject to our next requirement.

REQUIREMENT TE1: Web-based

The system must be web-based and accessible without the need to install additional browser plugins.

In [FAB⁺11b], Farwick et al. outline that a solution which facilitates the automated collection of information for an EA model must feature an internal data structure that can be understood by machines. Farwick et al. refer to the work of Tanner and Feridun [TFN09] whose general idea is to define declarative mappings to arbitrary information sources. Tanner and Feridun propose a loosely coupled approach for an integration of different information systems. They employ source specific queries to map information to an internal store and point out the importance of the implied semantics of imported information.

REQUIREMENT TE2: Internal Data Structure

The system must feature an internal data structure capable to store an entity its attributes and implied semantics.

4.4 Summary

In this chapter, we gave an overview of the characteristics of a federated EA model environment and detailed typical use cases of Federated EA Model Management. We employed the use cases and known challenges from literature and practice (cf. [HMR12]) to deduce requirements for a federal system and necessary process support.

In Table 4.15, we provide an overview of the use cases and outline which of the requirements address a use case. While most requirements were derived from literature or our empirical findings, some are inferred from the use cases or as a logical consequence when taking a usability perspective. Although important, in the present thesis, additional non-functional requirements are not considered.

Table 4.15: Mapping of use cases to requirements and respective categories

Requirement	Use Case										
	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11
PR1			○			○	●	●		●	
PR2			○			●	●	●	●	●	○
PR3		○				●	○	○	●	●	●
PR4	○	○									
Co1							●	●	○	○	●
Co2						●				●	
Co3							○	○			○
Co4							○	○		●	
Co5								●	○	○	●
Co6								●			●
Mo1	●	●	●	●		●				○	
Mo2			●	●							
Mo3	●	●	●	●	●	●					
Mo4				●	○						
Mo5		●									
Mo6			○	○	○	○	○	○	○		○
Mo7					●				○		
Mo8						●					
Mo9	●					●			●	○	
Mo10						●	●		●		
Mo11	○	●	●	●	●	●	●	●	●		
Mo12			●		●						
Mo13			●	●	●	○	●	●	●		
Mo14	●	●					○	●	●	●	●
Us1					●	○	●	●	●	●	●
Us2					●	○	●	●	●	●	●
Us3			●		●		●	●			●
Us4							●	●			
Us5							●	●			
Us6			○				●	●	○		
Us7						●	●	●			
Us8	○	○			●	○	●	●	●		○
TE1							○	○			○
TE2				○	○	○					

○ denotes a requirement that is not crucial to fulfill a use case but is meaningful for the use case either to increase usability or convenience of the user or other systems.

● denotes a requirement that is required and must be supported by software.

FEDERATED EA MODEL MANAGEMENT DESIGN

In this chapter, we detail a design of Federated EA Model Management. It is derived from the use case analysis in close collaboration with practitioners (cf. Chapter 7) which are currently pursuing an integration of federated information sources with a central, i.e. federal, EA model.

We start by further elaborating our conceptual picture that guides the present thesis. Then, we add a process model that outlines the high-level phases and their transitions within Federated EA Model Management. Further, model and metamodel conflicts are illuminated in a twofold manner, i.e. their detection as well as their resolution. For the latter, we provide visual means to anticipate and resolve conflicts. Visual differencing empowers users to perform a semi-automated quality assurance. During this step, enterprise architects often can anticipate possible conflicts that originate from differences in models as well as metamodels. A conflict management dashboard allows users to analyze arising model conflicts and to resolve conflicts in context of other model elements. Thereby, we emphasize (real-time) collaboration and the concept of tasks for asynchronous and synchronous conflict resolution.

5.1 System Design

We explain the design of Federated EA Model Management by elaborating our conceptual view of a federated EA model environment (cf. Figure 1.1 and Figure 4.1). Thereby, we detail concepts that are central for Federated EA Model Management. Then we introduce a metamodel that serves to build the foundation for the subsequent sections.

5. Federated EA Model Management Design

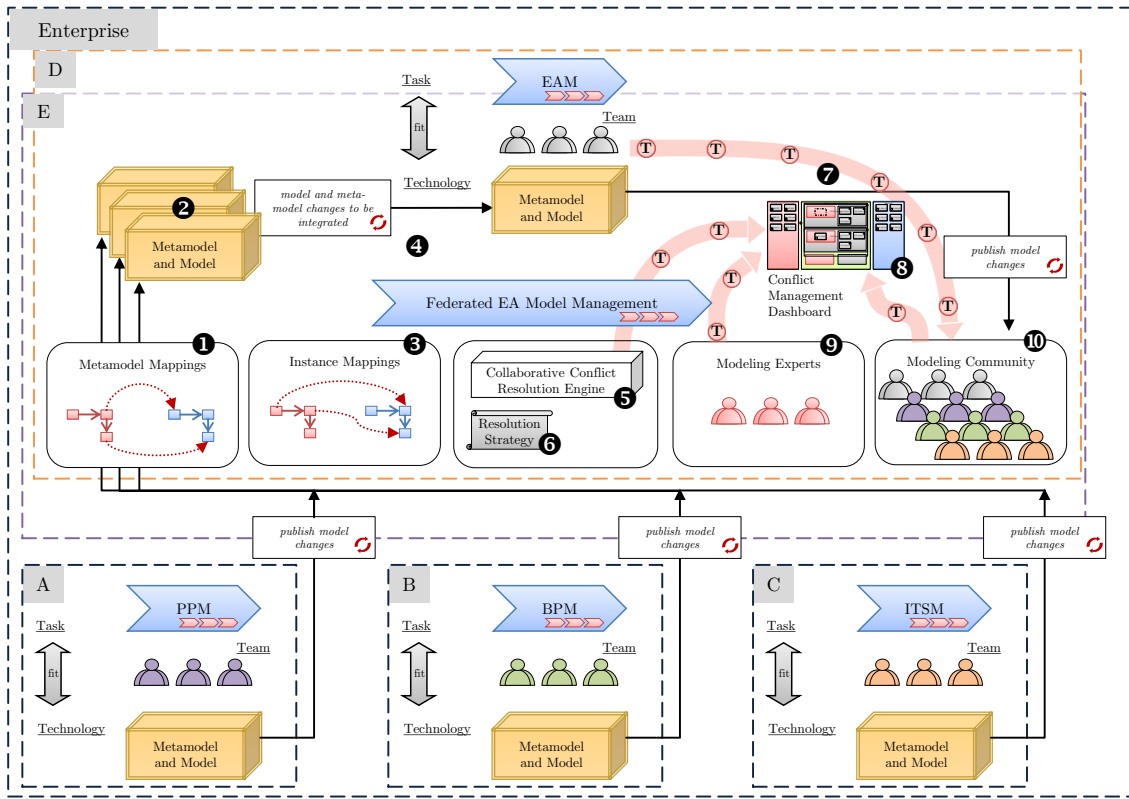


Figure 5.1: Conceptual overview of MODELGLUE: EA model management in a federated environment

5.1.1 Overview of MODELGLUE

We already detailed different aspects that are relevant for integration and subsequent synchronization of information sources with an EA model. Figure 5.1 illustrates a federated structure for information management in an organization pursuing EA management. Different modeling communities utilize an information source. The decentralized and autonomous modeling communities are denoted **A**, **B**, and **C** whereas a federal EA management function is denoted **E**.

In Figure 5.1, we add specifics relevant for Federated EA Model Management. In Federated EA Model Management, the parts of an information source's model and the respective import metamodel which are intended to be integrated with the EA model are kept locally as deep copies. This concept comes close to what in DWH is referred to as a 'staging-area', cf. [KC04, p. 31]. These local copies are under version control such that changes over time can be traced or analyzed. In line with Assumption 4.6, we exploit the ability of an EA repository to create and manipulate arbitrary metamodels. This way, in Figure 5.1 we further consider the federal system and the EA repository as one physical information system.

So far, **E** was viewed separately (cf. Figure 1.1 and Figure 4.1). In Figure 5.1 the communities **D** and **E** are merged, since

- EA management oversees and manages a federated EA model environment, hence takes an integral part of the responsibilities in the Federated EA Model Management process, and
- the federal system, MODELGLUE, and the EA repository can be viewed as a single information system.

MODELGLUE supports the creation of executable metamodel mappings (see **1** in Figure 5.1). In the course of an import of changes, information is stored in the local copies (see **2** in Figure 5.1) and instance mappings (see **3** in Figure 5.1) are created to reconcile the identity of a model element in case of model conflicts: an element can be traced back to its information source within the model community.

The local copies are import models that can be synchronized with the EA model (see **4** in Figure 5.1). Thereby, a collaborative conflict resolution engine (see **5** in Figure 5.1) facilitates the synchronization. It employs a conflict resolution strategy (see **6** in Figure 5.1) that details how MODELGLUE responds to a conflict, e.g. how conflicts are resolved automatically or how to communicate a conflict. As a result of this synchronization, which is performed as a long-lasting, collaborative merge of models, MODELGLUE generates conflict tasks (see **7** in Figure 5.1) which inform users about conflicts between different models and metamodels. The resolution of conflicts is facilitated by an interactive conflict management dashboard (see **8** in Figure 5.1). Although the modeling experts (see **9** in Figure 5.1) guide the resolution process and have special expertise in general modeling and EA modeling, the members of all modeling communities (**A**, **B**, and **C** in Figure 5.1) are involved in the conflict resolution. They are informed by tasks generated either manually by modeling experts or automatically by MODELGLUE. The tasks can be delegated to different members of the federated modeling community (see **10** in Figure 5.1). This community is the union of the members of the decentralized modeling communities (**A**, **B**, and **C** in Figure 5.1) as well as the EA Team.

5.1.2 A Metamodel for Evolutionary EA Modeling in a Federated Environment

In Section 2.2 we outlined the different layers (M_0 – M_3) as specified by the Object Management Group (OMG). Although we elaborate that we do not stick to this model strictly, we refer to the model presented subsequently as a metamodel. After introducing core concepts of the metamodel, we discuss this issue in the light of linguistic and ontological instantiation in Section 5.1.2.8.

Figure 5.2 presents a metamodel (m1) that fosters conflict resolution in a collaborative and federated EA model environment. The metamodel advances the approach of Neubert [Ne12] and incorporates extensions to his model. In contrast to Neubert, we abstract from the implementation as a wiki-based system. Central extensions relevant for Federated EA Model Management are (cf. also [RHM13a]):

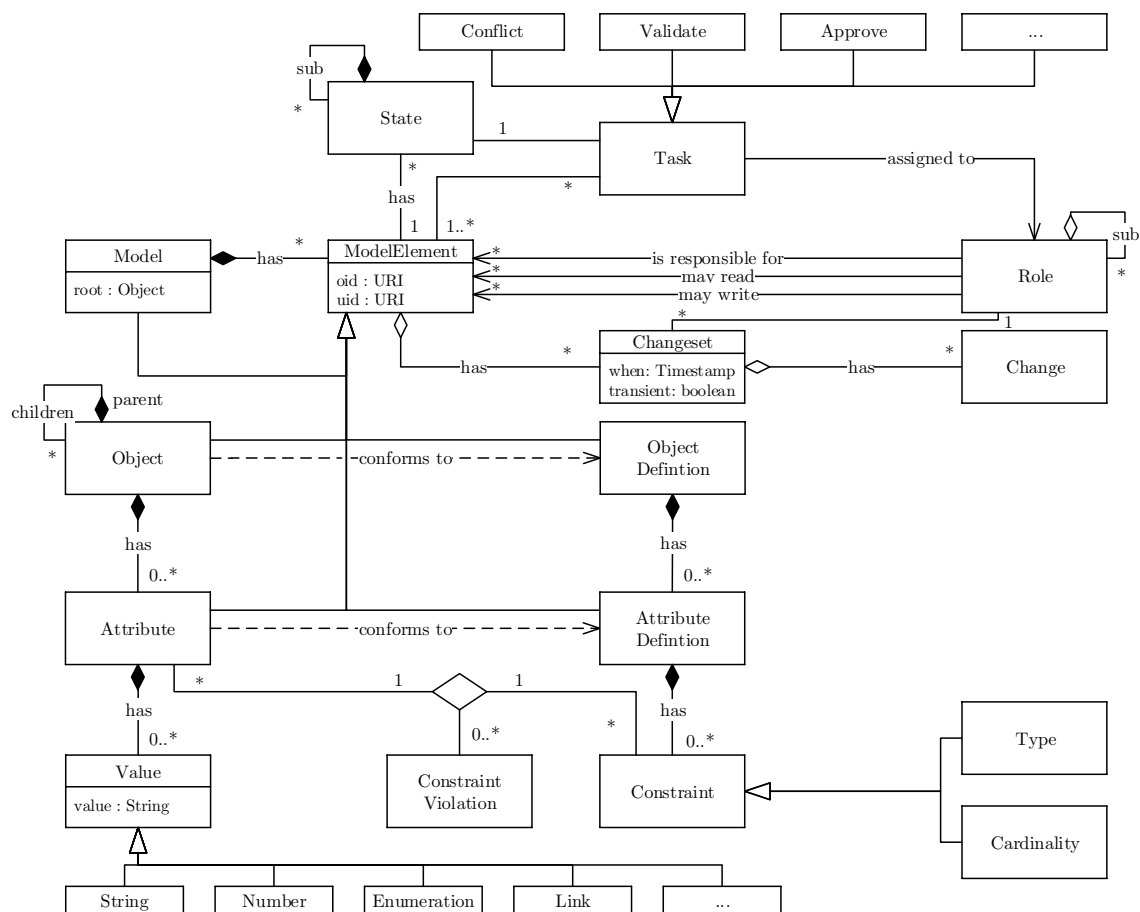


Figure 5.2: A language for Federated EA Model Management

- the notion of the abstract MODELEMENT and implied ROLE and STATE concepts,
- the separation between types, i.e. OBJECTDEFINITION and ATTRIBUTEDEFINITION, and instances, i.e. OBJECT and ATTRIBUTE,
- the introduction of a TASK and respective subclasses, and
- the transient CHANGESETS that can be attached to a TASK to capture model conflicts.

In the following, we detail features of this general purpose metamodel with respect to the application domain EA management. Thereby, we emphasize concepts that support a Federated EA Model Management.

5.1.2.1 Model Elements

Each MODELEMENT is identified via an immutable unique identifier (UID). Additionally, an origin identifier (OID) is used to inform the system about the origin of a MODELEMENT. This can be a UID within the system (from another model), or a UID within an external system. In the latter case, the first part of the UID, which ideally is implemented as an

URI, is a surrogate key that refers to the external system. Strictly speaking, surrogate keys carry no information other than to denote that a certain attribute exists [HOT76]. In contrast to the strict definition (cf. also [Da04, p. 434]), we use surrogate keys to refer not only to ATTRIBUTES but also to OBJECTS as well as ATTRIBUTEDEFINITIONS and OBJECTDEFINITIONS. On creation of a new MODELELEMENT within the system, both identifiers carry the same value, i.e. $OID=UID$. This is in line with the definitions of surrogate keys (cf. [Da04, p. 434]), since UIDs are uniquely identified and each new MODELELEMENT gets assigned a UID that has never been used within the system.

5.1.2.2 Roles

Similar to Farwick et al. [FPB⁺12], we attached roles to MODELELEMENTS such that our metamodel incorporates roles in a threefold manner. It realizes a fine-grained access control; besides explicit read and write access, responsibility is modeled as well. That means each MODELELEMENT may have different ROLES assigned to it. These ROLES are allowed to read/write content whereas others, usually a subset thereof, take over responsibility for a MODELELEMENT. Note that ROLES may be grouped such that a ROLE can be either a single person or an entire group possibly with subgroups. Further, a ROLE can also be an information system.

In an EA model, it should be possible to define concepts, i.e. models or objects, for which responsibility is partially shared in terms of more fine-grained responsibilities (cf. Example 5.1).



EXAMPLE 5.1: Different responsible roles for a single object

An OBJECT with OBJECTDEFINITION ‘application’ has an ATTRIBUTEDEFINITION ‘uptime’; an instance of such an application, e.g. SAP CRM, has an application owner assigned as responsible ROLE for this OBJECT. However, for maintenance reasons, the responsibility of the ATTRIBUTE ‘uptime’ is delegated to the respective system administrator since this ROLE is also responsible for installing patches, server restarts, etc.



In contrast to Neubert [Ne12] and through the introduction of the MODELELEMENT, we define access rights and responsibilities on instance level for MODELS, OBJECTS, and ATTRIBUTES as well as on type level for OBJECTDEFINITIONS and ATTRIBUTEDEFINITIONS. This not only offers a more fine-grained access control as well as more precision when modeling responsibilities, but allows a logical inheritance of responsibility. We explain the separation of types and instances and come back to this concept of inheritance of responsibility afterwards.

5.1.2.3 Separation of Types and Instances

In line with Neubert [Ne12], our approach offers flexibility as ATTRIBUTES may or may not conform to an ATTRIBUTEDEFINITION. Due to a separation of the concepts ATTRIBUTE,

ATTRIBUTEDEFINITION, and VALUE, end-users are free to store VALUES that do not conform to their definition, which is commonly perceived as a larger degree of freedom in an early modeling phase [MNS12]. That means, inconsistent states of the model in terms of conformance to the respective definitions are tolerated by the model and end-users may maintain OBJECTS and ATTRIBUTES regardless of their exact type conformance. This fosters a bottom-up approach as described by [Ne12, p.41ff]. Neubert calls the explicit specification of an ATTRIBUTE type within an ATTRIBUTEDEFINITION ‘*type constraint*’. In our metamodel, we denote this concept with the TYPE subclass of CONSTRAINT.



EXAMPLE 5.2: Benefits through an increased degree of freedom

An OBJECT, SAP CRM, with OBJECTDEFINITION ‘Application’ has an attribute ‘service level’. Within its ATTRIBUTEDEFINITION, the ATTRIBUTE is defined as an enumeration {gold, silver}. An information source uses the additional service level platinum, i.e. a new enumeration element for ‘service level’ is required. The problem just has been identified and the organization did not yet decide whether to treat platinum levels differently. However, the system allows end-users to add new VALUES regardless of their type, stores the VALUE, and informs the end-user about the type violation. Meanwhile, all information is maintained and the employees that are familiar with the platinum level are aware of its implications.



Example 5.2 illustrates the benefits of loose coupling between data (EA model) and schema (EA metamodel). We call an ATTRIBUTE that has an ATTRIBUTEDEFINITION which enforces conformance *strict*. By default, end-users may add ATTRIBUTES relevant for their particular purposes without any ATTRIBUTEDEFINITION. In the EA domain, the EA repository manager may add an ATTRIBUTEDEFINITION based on the frequency an ATTRIBUTE has been used or the number of occurred type violations. The idea is not to maintain inconsistency, but to temporarily allow inconsistencies that are consolidated in the long run. This brings us back to the discussion about the responsibility of a particular model element.

5.1.2.4 Inheritance of Responsibility for Model Elements

Algorithm 1 can be used to determine the responsible role of an ATTRIBUTE with the chain of responsibility {Model, ObjectDefinition, Object, AttributeDefinition, Attribute}. This chain gets shortened for determining the responsible role of an ATTRIBUTEDEFINITION, {Model, ObjectDefinition, AttributeDefinition}. The same holds true for an OBJECT, i.e. {Model, ObjectDefinition, Object}, and the OBJECTDEFINITION, {Model, ObjectDefinition}. For the MODEL, responsibility must be configured on creation such that it can always be determined directly and does not need to be derived from other MODELELEMENTS. Since the MODEL is the last escalation instance, it is mandatory to set a responsible role for a MODEL. The escalation path is traversed from finest level on

instances through types and then to the next level of instances and types up to the model. This way, the fine-grained access control can be used, but, however, more coarse-grained access control concepts can also be enforced with respect to the chain of responsibility as given above. Thereby, σ denotes a utility function that is able to query the respective property of a MODELELEMENT whose classifier is indicated by the current position in the chain.

Algorithm 1: Determining responsible roles of an attribute

```

1: Input : chain of responsibility as stack, model element  $\mathcal{E}$ 
2: Output: responsible role  $\rho = \emptyset$ 
3: function determineResponsibleRole(chain,  $\mathcal{E}$ ):
4:    $\mathcal{E}_{current} \leftarrow \mathcal{E}$ 
5:   while  $\rho \equiv \emptyset$  do
6:      $\mathcal{E}_{current} \leftarrow \sigma(\text{chain.pop}(), \mathcal{E}_{current})$ 
7:     if  $\exists \mathcal{E}_{current}.\text{responsibleRole}$  then
8:        $\rho \leftarrow \mathcal{E}_{current}.\text{responsibleRole}$ 

```

5.1.2.5 Tasks and States

We introduce the notion of TASKS to facilitate the semi-automated model maintenance within MODELGLUE in order to keep the model consistent during its evolution. These tasks are meant to modify the STATE of a MODELELEMENT. Before we detail the STATES of a TASK and the different kinds of TASKS, we put focus on the nature of the MODELELEMENT and its life-cycle with respect to model conflicts.

Figure 5.3 illustrates the different STATES of a MODELELEMENT as well as transitions between these states. Initially, each MODELELEMENT is in STATE *normal*. In his Master’s Thesis, Kirschner [Ki14, p.27] calls this STATE ‘*clean*’, since no pending changes are to be applied to a MODELELEMENT. For our next considerations, we assume a MODELELEMENT is involved in a conflict with concurrent modifications on another or the very same MODELELEMENT. The STATE *in conflict* is devoted to this condition. The general idea behind the STATE transitions of a MODELELEMENT is that the MODELELEMENT is in conflict if TASKS for this element exist, but are not yet viewed or opened by the assigned responsible ROLE. These TASKS are a result of the conflict detection in the course of a model merge. In subsequent sections, we point out how this detection is performed. As soon as a TASK is not in STATE *new* (cf. Figure 5.4), the MODELELEMENT is in STATE *under consolidation*. To some extent, the aggregation of ATTRIBUTES to an OBJECT and ATTRIBUTEDEFINITIONS to OBJECTDEFINITION denotes a part-whole relationship. This semantic part-whole relationship has an impact on the determination of the STATE of a MODELELEMENT. In the following we describe the STATES of different MODELELEMENTS more formally.

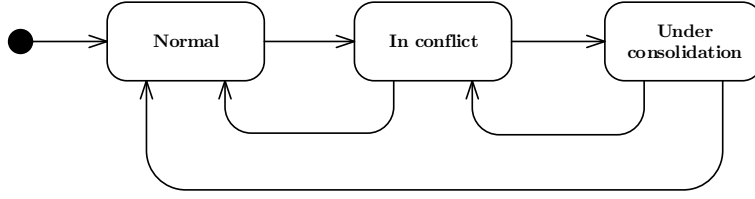


Figure 5.3: States of a MODELELEMENT

Let \mathbb{T} be the set of all instances of TASK within MODELGLUE. Then the STATE $S^{\mathcal{M}}$ of a MODEL \mathcal{M} is determined by Equation 5.1.

$$S^{\mathcal{M}} = \begin{cases} \text{in conflict} & \text{if } \exists t \in \mathbb{T}(t.modelElement \in \mathcal{M} \wedge \\ & t.state = \text{"unresolved"} \wedge t.state.sub = \text{"new"}) \\ \text{under consolidation} & \text{if } \forall t \in \mathbb{T}(t.modelElement \in \mathcal{M} \wedge \\ & t.state = \text{"unresolved"} \wedge t.state.sub \neq \text{"new"}) \\ \text{normal} & \text{else} \end{cases} \quad (5.1)$$

Through the generalization of ATTRIBUTE, OBJECT, OBJECTDEFINITION and ATTRIBUTEDEFINITION to a MODELELEMENT, TASKS that address these concepts are included in Equation 5.1.

Further, let \mathcal{O} be an OBJECT with its ATTRIBUTES $\mathcal{A}_{1..n}$, then the state $S^{\mathcal{O}}$ of \mathcal{O} is calculated as described in Equation 5.2.

$$S^{\mathcal{O}} = \begin{cases} \text{in conflict} & \text{if } \exists t \in \mathbb{T}(t.modelElement = \mathcal{O} \vee t.modelElement \in \mathcal{A}_{1..n} \wedge \\ & t.state = \text{"unresolved"} \wedge t.state.sub = \text{"new"}) \\ \text{under consolidation} & \text{if } \forall t \in \mathbb{T}((t.modelElement = \mathcal{O} \vee t.modelElement \in \mathcal{A}_{1..n}) \wedge \\ & t.state = \text{"unresolved"} \wedge t.state.sub \neq \text{"new"}) \\ \text{normal} & \text{else} \end{cases} \quad (5.2)$$

For a single attribute \mathcal{A} the state $S^{\mathcal{A}}$ is given by Equation 5.3.

$$S^{\mathcal{A}} = \begin{cases} \text{in conflict} & \text{if } \exists t \in \mathbb{T}(t.modelElement = \mathcal{A} \wedge \\ & t.state = \text{"unresolved"} \wedge t.state.sub = \text{"new"}) \\ \text{under consolidation} & \text{if } \forall t \in \mathbb{T}(t.modelElement = \mathcal{A} \wedge \\ & t.state = \text{"unresolved"} \wedge t.state.sub \neq \text{"new"}) \\ \text{normal} & \text{else} \end{cases} \quad (5.3)$$

The state $S^{\mathcal{D}^{\mathcal{O}}}$ of an OBJECTDEFINITION $\mathcal{D}^{\mathcal{O}}$ with the ATTRIBUTEDDEFINITIONS $\mathcal{D}_{0..n}^{\mathcal{A}}$ is calculated in a similar way to the state of an OBJECT as depicted in Equation 5.4.

$$S^{\mathcal{D}^{\mathcal{O}}} = \begin{cases} \text{in conflict} & \text{if } \exists t \in \mathbb{T}((t.modelElement = \mathcal{D}^{\mathcal{O}} \vee t.modelElement \in \mathcal{D}_{0..n}^{\mathcal{A}}) \wedge \\ & t.state = \text{"unresolved"} \wedge t.state.sub = \text{"new"}) \\ \text{under consolidation} & \text{if } \forall t \in \mathbb{T}((t.modelElement = \mathcal{D}^{\mathcal{O}} \vee t.modelElement \in \mathcal{D}_{0..n}^{\mathcal{A}}) \wedge \\ & t.state = \text{"unresolved"} \wedge t.state.sub \neq \text{"new"}) \\ \text{normal} & \text{else} \end{cases} \quad (5.4)$$

Finally, the state $S^{\mathcal{D}^{\mathcal{A}}}$ of an ATTRIBUTEDDEFINITION $\mathcal{D}^{\mathcal{A}}$ is given by Equation 5.5.

$$S^{\mathcal{D}^{\mathcal{A}}} = \begin{cases} \text{in conflict} & \text{if } \exists t \in \mathbb{T}(t.modelElement = \mathcal{D}^{\mathcal{A}} \wedge \\ & t.state = \text{"unresolved"} \wedge t.state.sub = \text{"new"}) \\ \text{under consolidation} & \text{if } \forall t \in \mathbb{T}(t.modelElement = \mathcal{D}^{\mathcal{A}} \wedge \\ & t.state = \text{"unresolved"} \wedge t.state.sub \neq \text{"new"}) \\ \text{normal} & \text{else} \end{cases} \quad (5.5)$$

Before we introduce the subclasses of TASK which implement specialized model maintenance tasks in Section 5.2.5.5, we elaborate on the general STATES of a TASK. Figure 5.4 provides an overview and the transitions between the STATES of a TASK. The STATES are explained in the following.

Unresolved denotes the initial state that is divided in further sub-states.

New is the initial sub-state a TASK is in. The TASK has been assigned to a ROLE, but not yet viewed by persons that are in possession of this ROLE and now responsible for this TASK.

Overdue denotes a TASK that is past its due date, i.e. each TASK has a desired deadline when it has to be either *resolved* or *ignored*. The transition to this state is performed by the system based on the due date.

Reviewed refers to a TASK that has been viewed by the responsible role, i.e. it is ensured a human being has been informed about the content of a TASK. Its resolution can be deferred and it can be forwarded to another ROLE such that the state is altered from *reviewed* to *new* again.

Ignored means the TASK actively has been put to the ignored state. In this state the conflict resides within the system but parties agree to maintain an inconsistent state. This decision is remembered by keeping the instance in this state such that a subsequent conflict that refers to the same issue can be discarded (cf. Section 5.2.5.4).

Resolved is the final state and denotes that the related model conflict has been resolved. This state must be set manually.

From a user perspective, tasks can be either forwarded, ignored, or resolved to finish the piece of work denoted by an instance of a TASK.

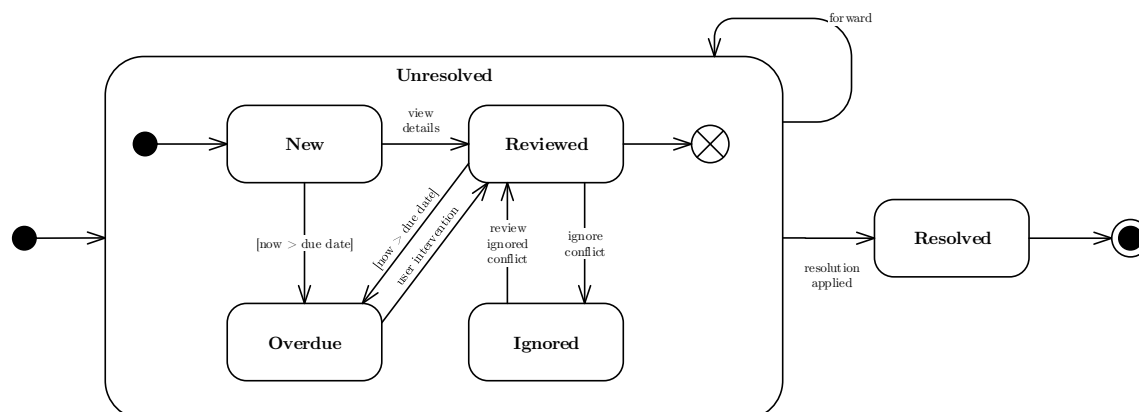


Figure 5.4: States of a TASK

One of our EA practitioners provided us with a vivid example for a typical situation in an organization in which ignoring model conflicts is indispensable:



EXAMPLE 5.3: Ignoring a model conflict permanently

Our accounting systems do not allow any changes once an annual report is published. Thus, there is no other solution for any conflict with these systems then to ignore the conflict and ‘live with it’.



5.1.2.6 Transient Changesets

CHANGESETS of a MODELELEMENT are used to keep track of recent changes. They contain the respective role that initiated the transaction and a timestamp. Through the connection of MODELELEMENTS and CHANGESETS as well as the relationship to the abstract concept of TASKS, the concrete TASKS may refer to OBJECTS, OBJECTDEFINITIONS, ATTRIBUTES, and ATTRIBUTEDEFINITIONS. Note that possible model conflicts are stored within subclasses of a TASK including references to involved CHANGESETS¹.

On the one hand, CHANGESETS save a history of operations that have been applied by a user. This way, they allow to track changes over time and can serve as audit trails. On the other hand, not yet applied changes are stored in CHANGESETS that are marked as *transient*, i.e. they may or may not get applied to an element.

In MODELGLUE, CHANGESETS are realized as a delta-backward version history. In Section 2.3.9 we conclude that such an approach has the advantage of a fast access to the most recent version and moreover saves a considerable amount of storage compared to a fully materialized version history.

¹For a detailed description how to save model changes as patches, we refer the interested reader to Kelter et al. [KKK13] and Kirschner [Ki14, p. 24].

5.1.2.7 Constraints and Constraint Violations

An OBJECTDEFINITION is defined by its name that is unique within a model, a comprehensive and meaningful description of its semantics, and through its ATTRIBUTEDEFINITIONS. As outlined above, ATTRIBUTEDEFINITIONS can be equipped with CONSTRAINTS. For Federated EA Model Management, we use two kinds of constraints. These are (cf. [Ne12, p. 33]):

- Type constraints and
- Cardinality constraints.

In [Ne12, p. 35], Neubert discusses the importance not only to validate a model on demand, i.e. on access, but also to persist the validation status. In [Ne12, p. 101], he discusses an efficient technique to persist the validity of an ATTRIBUTE. In addition to Neubert, we propose to persist violations of constraints as first-class objects. CONSTRAINTVIOLATIONS give information about the current conformity of a model to its metamodel. For an EA model this conformity serves as a quality indicator (cf. Chapter 7).

5.1.2.8 Excursion: Linguistic vs. Ontological Instantiation

In [Kü06], Kühne investigates the role of different models and distinguishes between the ‘*token model*’ and the ‘*type model*’. The token model “holds between a system and a model representing the former in a one-to-one fashion. Model elements are regarded as designators for system elements”[Kü06]. The ‘*token model-of*’ relationship is denoted \triangleleft_i . The type model “holds between a system and a model classifying the former in a many-to-one fashion. Model elements are regarded as classifiers for system elements”[Kü06]. The ‘*type model-of*’ relationship is denoted \triangleleft_t . In his discourse, Kühne further distinguishes ‘*linguistic instantiation*’ denoted \triangleleft^l and ‘*ontological instantiation*’ denoted \triangleleft^o (cf. Definition 8 and 10 in [Kü06]). We employ these foundations to describe the nature of our model that we so far referred to as a metamodel.

Let us assume an EA as a system of systems denoted \mathcal{S} and an EA model \mathcal{M}_{EA} with an EA metamodel \mathcal{MM}_{EA} . Then, the following relationships between these models hold true:

On the one hand, \mathcal{M}_{EA} is a token model of \mathcal{S} .

$$\mathcal{S} \triangleleft_i \mathcal{M}_{EA} \quad (5.6)$$

On the other hand, the EA metamodel is an ontological token model of the EA model.

$$\mathcal{M}_{EA} \triangleleft_t^o \mathcal{MM}_{EA} \quad (5.7)$$

Both, the EA model and its metamodel are a linguistic *instanceOf* (cf. also [AK03]) a common metamodel.

$$\mathcal{M}_{EA} \triangleleft_t^l \mathcal{L} \wedge \mathcal{MM}_{EA} \triangleleft_t^l \mathcal{L} \quad (5.8)$$

In its classical sense, \mathcal{L} could be considered to be a meta-metamodel since we intend to use it as a means to specify EA metamodels, hence each EA metamodel is an instanceOf a meta-metamodel. On the other hand, \mathcal{L} is also a metamodel since the EA model is also an instanceOf \mathcal{L} . To some extent, our model \mathcal{L} is level agnostic since it incorporates concepts of different levels (cf. Figure 2.7 on p. 29).

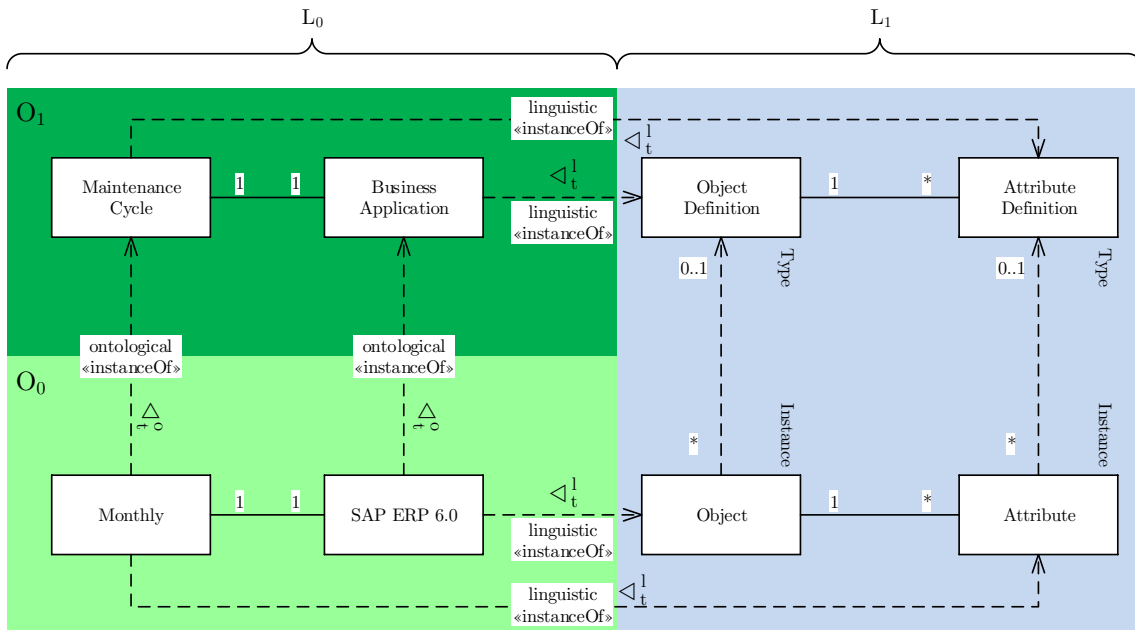


Figure 5.5: Linguistic and ontological instantiation of core concepts in our metamodel based on Kühne [Kü06]

We employ Kühne’s notation to illustrate the different instantiation types in Figure 5.5. In the remainder of the present thesis, we proceed referring to this model as a metamodel. We further assume that the metamodel of an EA model is an linguistic instance of language \mathcal{L} which is defined by the metamodel illustrated in Figure 5.2.

5.2 Process Design

In this section, we outline the core activities of Federated EA Model Management. First, we present an iterative and incremental process design for Federated EA Model Management. Thereafter, we establish our understanding of a federated EA model environment as branches of a federal model. In the subsections of this section, we present details of the process and elaborate the design and techniques of MODELGLUE. In line with the OMG, we perceive a process as structured guide for certain roles to achieve particular goals. Processes are subdivided in *tasks* employing context information and eventually utilizing tools to create artifacts, i.e. tasks describe the essential activities that have to be performed within a process. In line with the Business Process Modeling Notation (BPMN) [Ob10], we define a task as follows.

DEFINITION 5.1: Task

A task denotes a single piece of work performed by a role to execute an explicit or implicit process. ■

Figure 5.6 shows an iterative process to continuously integrate models from specialized and autonomous modeling communities within an enterprise into a holistic and consistent EA model. We briefly describe the activities which correspond to the use cases introduced in Section 4.2 and outline how they are interrelated.

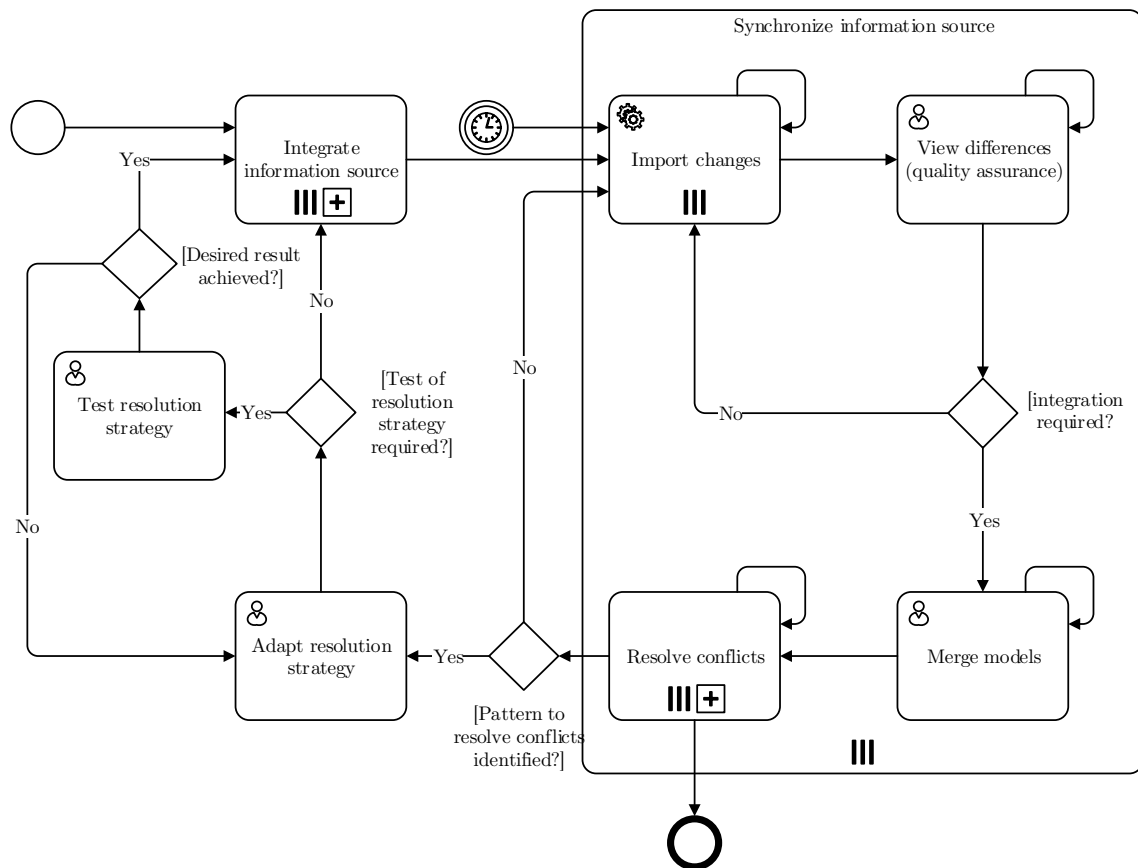


Figure 5.6: Process model for integrating information sources in a federation

Integrate information source includes the identification of relevant information sources, the alignment of terminology, and the logical as well as physical definitions of mappings from an information source to the EA model. It is divided in additional sub tasks and can be executed in parallel.

Synchronize information source is an iterative sub-process that seeks to import information to a staging area and subsequently merge changes of the model to an EA model.

Import changes describes the mere import of information from a source to a model within MODELGLUE. The task is performed entirely automated and is either time-triggered or triggered in the event of a synchronization.

View model differences serves as a quality assurance step; thus, it is performed by a human being. The EA repository manager views at the model differences and decides to continue after sampling some well-known MODELELEMENTS for their correctness. Thereby, the decision to continue with the merge of models is more based on experience with the involved models and ‘gut feeling’ than on a comprehensive analysis.

Merge models denotes the process to take over model changes of an information source’s model to the EA model.

Resolve conflicts summarizes the (collaborative) activities to find a resolution to model inconsistencies of any kind including model as well as metamodel conflicts.

Adapt resolution strategy is the reactive or proactive definition of a strategy to automate conflict resolution. This automation can incorporate notifications via human tasks such that it represents a de facto (semi-)automated resolution process.

Test resolution strategy serves to test the newly created or adapted resolution strategy. This is commonly done with a subset of actual live information rather than purely synthetic information.

Note that Figure 5.6 represents the core process and tasks of Federated EA Model Management and is not meant as a comprehensive model with all possible transitions. For instance, during the quality assurance step it could turn out that the import delivered flawed or corrupted information. Possible counter actions may include contacting the data owner such that recent development changes and any undesired side-effects are resolved collaboratively.

5.2.1 A Federated Model Environment as Branches of a Federal Model

In Section 4.1.3 we already explained the observed phenomena of semantically overlapping models within an enterprise. In the following, we view at the different models of information sources as branches of an EA model. This can be accomplished by creating a branch \mathcal{M}^a of the current state of the EA model \mathcal{M}^b .

In line with Definition 5.2, we consider \mathcal{M}^a a branch of \mathcal{M}^b . \mathcal{M}^a serves as staging model whereas the EA model \mathcal{M}^b co-evolves independently. This staging model serves as an import model for one information source, i.e. for n information sources one intends to integrate in the federation, n branches of the EA model are created such that \mathcal{M}^b and branches $\mathcal{M}_{1..n}^a$ build a federation $\mathcal{M}_{1..n}^a \overset{\circ}{\mapsto} \mathcal{M}^b$ (cf. Definition 4.5 on p. 81).

DEFINITION 5.2: Branch

A model element \mathcal{E}^a is a branch of \mathcal{E}^b iff

$$\mathcal{E}^{a \rightarrow b} \Leftrightarrow \exists \mathcal{E}^b (\mathcal{E}^a.oid \equiv \mathcal{E}^b.oid \wedge \mathcal{E}^a.uid \neq \mathcal{E}^b.uid)$$

A model \mathcal{M}^a is a branch of a model \mathcal{M}^b iff

$$\mathcal{M}^{a \rightarrow b} \Leftrightarrow \exists \mathcal{E}^a \in \mathcal{M}^a \exists \mathcal{E}^b \in \mathcal{M}^b (\mathcal{E}^{a \rightarrow b})$$

■

Such a branch creates a clone of the EA model that references its origin. This clone not only includes a deep copy² of a model but also includes its revisions (versions). Thereby, these newly created model elements have different UIDs than their origin. The OID serves as surrogate to mark the origin of a particular model element by storing a reference to the original UID. Algorithm 2 describes this branching more formally. Thereby, let ϕ be a utility function that creates UIDs.

Algorithm 2: Branching a model in MODELGLUE

```

1: function branch( $\mathcal{E}_s, \mathcal{E}_t$ ):
2:    $\mathcal{E}^*$  = new instance
3:    $\mathcal{E}^*.uid = \phi$ 
4:    $\mathcal{E}^*.oid = \mathcal{E}_s.oid$ 
5:   // copy changesets
6:   // copy access rights
7:   if  $\mathcal{E}_t \neq \emptyset$  then
8:      $\mathcal{E}_t = \mathcal{E}^* \cup \mathcal{E}_t$  // includes models, attributes, objects and their
       definitions
9:   // copy sub-model elements
10:  foreach  $\mathcal{E}_i \in \mathcal{E}_s$  do
11:     $\text{branch}(\mathcal{E}_i, \mathcal{E}^*)$  // recursion

```

Before we proceed with the further steps that are required to integrate an information source, we provide rationale for our notion of the OID. Kirschner [Ki14] proposes to use a list of OIDs. At a first glance, it looks like an entire collection of OIDs is needed for each MODELELEMENT in particular when allowing to merge multiple models with one model. We argue, that an OBJECT denotes only the name and order of ATTRIBUTES. The same holds true for a model which serves as a named container for OBJECTS and OBJECTDEFINITIONS. A more fine-grained perspective on an OBJECT as an ordered collection of ATTRIBUTES whose elements in turn may have different OIDs than their OBJECT reveals that a list of OIDs is not necessary to keep track of different origins of an OBJECT and its ATTRIBUTES. One has to look at the ATTRIBUTES and OBJECTS and respective definitions as separate entities. Since all these MODELELEMENTS are equipped with separate OIDs and UIDs, OBJECTS, OBJECTDEFINITIONS as well as MODELS are able to aggregate information

²Note that literature often refers to such a deep copy as a fork.

that originates from different information sources. This allows for OBJECTS to merge ATTRIBUTES of different branches, i.e. each ATTRIBUTE may refer to different branches whereas the OBJECT's name originates from exactly one branch. A change of an OBJECT's OID is determined by the origin of its name and attribute order as well as its description. In addition, meta-information of each merge action, e.g. the user that triggered the merge and an exact timestamp, is stored within the system (see [Ki14, p. 28ff]).

After this discourse about the OID, we outline how a federation can be viewed as branches of models. This is meant to give an overview of the general principle that is refined in subsequent sections. During the task **integrate information source** for each information source, a branch is created. More formally, we can write $\mathcal{M}_{a_{1..n}} \xrightarrow{\circ} \mathcal{M}_b \wedge \forall \mathcal{M}^x \in \mathcal{M}_{a_{1..n}} (\mathcal{M}^{x \rightarrow b})$ whereas \mathcal{M}_b denotes the EA model and $\mathcal{M}_{a_{1..n}}$ denote the import models of information sources. As a side effect, all import models $\mathcal{M}_{a_{1..n}}$ carry the OID of the EA model. These branches $\mathcal{M}_{a_{1..n}}$ not only serve as import model but also as a target model and metamodel for the mapping that is to be defined subsequently (cf. Section 5.2.2.5). Source models and metamodels are considered the export models of the respective information sources. During this mapping, the creation of OIDs that serve as surrogates to refer to information sources is defined. Commonly, the URI of the information source is incorporated with the identifier of a MODELELEMENT in the information source. During the process step **import changes**, changes to OBJECTDEFINITIONS and ATTRIBUTEDEFINITIONS as well as changes to existing OBJECTS and ATTRIBUTES can be traced to their origin. Importing changes from information sources is triggered whereas OIDs serve as surrogate for the involved information sources. At this point, the MODELELEMENTS of an import model commonly consist of OIDs that point to the EA model as well as OIDs that point to information sources. The models $\mathcal{M}_{a_{1..n}}$ and \mathcal{M}_b are merged and arising conflicts are detected (task **merge models**). A preview model contains the tentative merge result and serves to **resolve conflicts**. This commonly involved multiple parties and takes place in a collaborative fashion. Note that we propose an *incremental* extension to the metamodel of \mathcal{M}_b according to new information source that are to be integrated. Process further is *iterative* since changes of $\mathcal{M}_{a_{1..n}}$ on a frequent basis are imported and merged with the EA model \mathcal{M}_b .

5.2.2 Integrate Information Sources

Before we describe the sub-process which details how to integrate an information source, we discuss our notion of *integration*. In [Fr08], Frank proposes a conception of integration. In line with Frank, Kattenstroth et al. [KFH13] present further insights on the conception of integration. We employ their work to take a closer look at the notion of integration in the context of MODELGLUE. Thereby, we assume that the reader is familiar with [Fr08] and [KFH13].

Figure 5.7 builds on the topic map presented by Kattenstroth et al. [KFH13]. We add additional dimensions found in [Fr08] and use Harvey Balls to denote to which extent our approach addresses integration. MODELGLUE first requires to define a unidirectional mapping (● meta-level) which operates on a shared understanding of concepts (● static). Our approach uses an export model to realize loose coupling (● coupling) and imports information that is merged to a common EA model (● merging). Via an OID, identity of

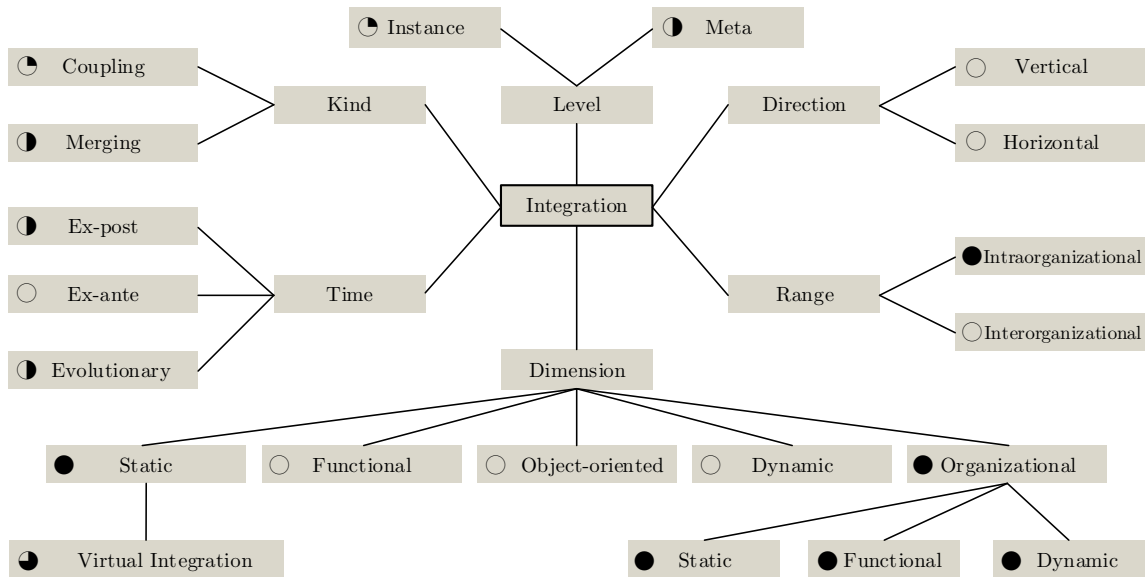


Figure 5.7: Topic map on the term integration and the perspectives taken in our approach

a piece of information can be reconciled (☉ instance). An information source commonly is a legacy system (☉ ex-post) whereas the integration can also take place throughout the evolution of an information system or the EA repository, because new concepts could address future concerns of stakeholders (☉ evolutionary). To a certain extent, we expect that information within the EA repository is also shared with other communities, but primarily the integration serves to procure information for the maintenance of the EA model (● virtual integration). Although we regard our process design as flexible in the way transitions are performed, the core activities need to be integrated into organizational processes (● organizational). That includes a static integration as well as functional and dynamic integration (● respectively). In our design, we limit the scope to models within one organization (● intraorganizational).

So far, we established an understanding of our notion of integration. Figure 5.8 describes the sub-process of the task *integrate information sources* as a BPMN diagram. As a first step towards an integration of an information source, one has to select a relevant information source. The selection of an information source is done by the EA coordinator. This selection depends on many factors such as relevant scope of model, quality, etc. such that the selection of an appropriate information source can be considered a challenging task. Especially the scope depends on a particular concern; a comprehensive list of typical concerns in EA management presented in seven categories can be found in [BEL⁺08, ch. 3]. In Section 4.1.2, we already reported empirical research results on information sources used in industry.

After an identification of the relevant part of the information source's metamodel that is intended to be integrated with the EA model, the EA coordinator informs the data owner that the information source is of particular interest for the EA model. The specific part must be detailed since commonly only some concepts within an information source are relevant for EA management (cf. Section 2.1.2). The data owner specifies the export metamodel of an information source. It is the task of the data owner to capture concepts

that have been requested by the EA coordinator. This export metamodel serves as a basis for the mapping to the EA repository.

In a next step, the EA modeling expert checks the quality of the information source's model as well as metamodel. Assuming that no further changes to the information source are necessary, the creation of the import metamodel is the next task. With our present understanding to view at a federation as branches of an EA model, the EA repository manager is instructed by the EA modeling expert to adapt the metamodel of the EA model such that concepts therein reflect the part of a new information source intended to import. That is, the activity *create import metamodel* also refers to the creation of an extension of the EA model. Depending on the part of an information source that has to be integrated with the EA model, considerable adaptations might be necessary. This step further includes the creation of a model that serves as a staging area for a particular information source, i.e. the import model and its metamodel. This is created by branching the extended EA model.

Next, the EA modeling expert, the EA repository manager, and the data owner develop a mapping between this import metamodel and the export metamodel. Exact semantics of concepts can be discussed and clarified best in a face-to-face workshop. As we detail below, once the models and semantics of their concepts are clarified, a conceptual mapping is created that serves as input for the EA repository manager who is tasked to create the physical, i.e. executable, model mapping. In this vein, the EA repository manager can also incorporate known resolution patterns for conflicts in the conflict resolution strategy. After this step, the actual integration is finished. However, most commonly, the EA repository manager triggers an initial synchronization with the information source.

5.2.2.1 Initial Considerations for Creating a Mapping for an Information Source

Interoperability and integration of information sources is a common problem of information systems that has been around for decades. To integrate multiple DBMSs, Spaccapietra et al. [SPD92] refer to these model mappings as '*schema integration*'. The authors outline the importance of an '*Generic Data Model (GDM)*' that serves as a common denominator.

In [No04] Noy discusses the conceptual difference of traditional information integration and semantic integration of ontologies. She highlights that in traditional information integration, models of different information sources exist prior to the common schema. The schema used for an integration is "only general enough to provide access to all the schemas that it integrates" [No04]. In contrast to traditional schema mapping approaches, the vision of ontologies is that an upper ontology exists which serves as a frame of reference. It is assumed that all modelers agree upon this reference ontology prior to creating extensions thereof that detail specific aspects. These extensions are commonly called domain ontologies. Some approaches propose an automated mapping employing an upper ontology to map different domain ontologies, see e.g. [SE13].

Although ontologies have advantages, e.g. support for reasoning engines, in line with Noy [No04], we stress that as of today automatic mapping between ontologies is not feasible considering today's enterprise IT environments. Practitioners (cf. Chapter 7) confirm that

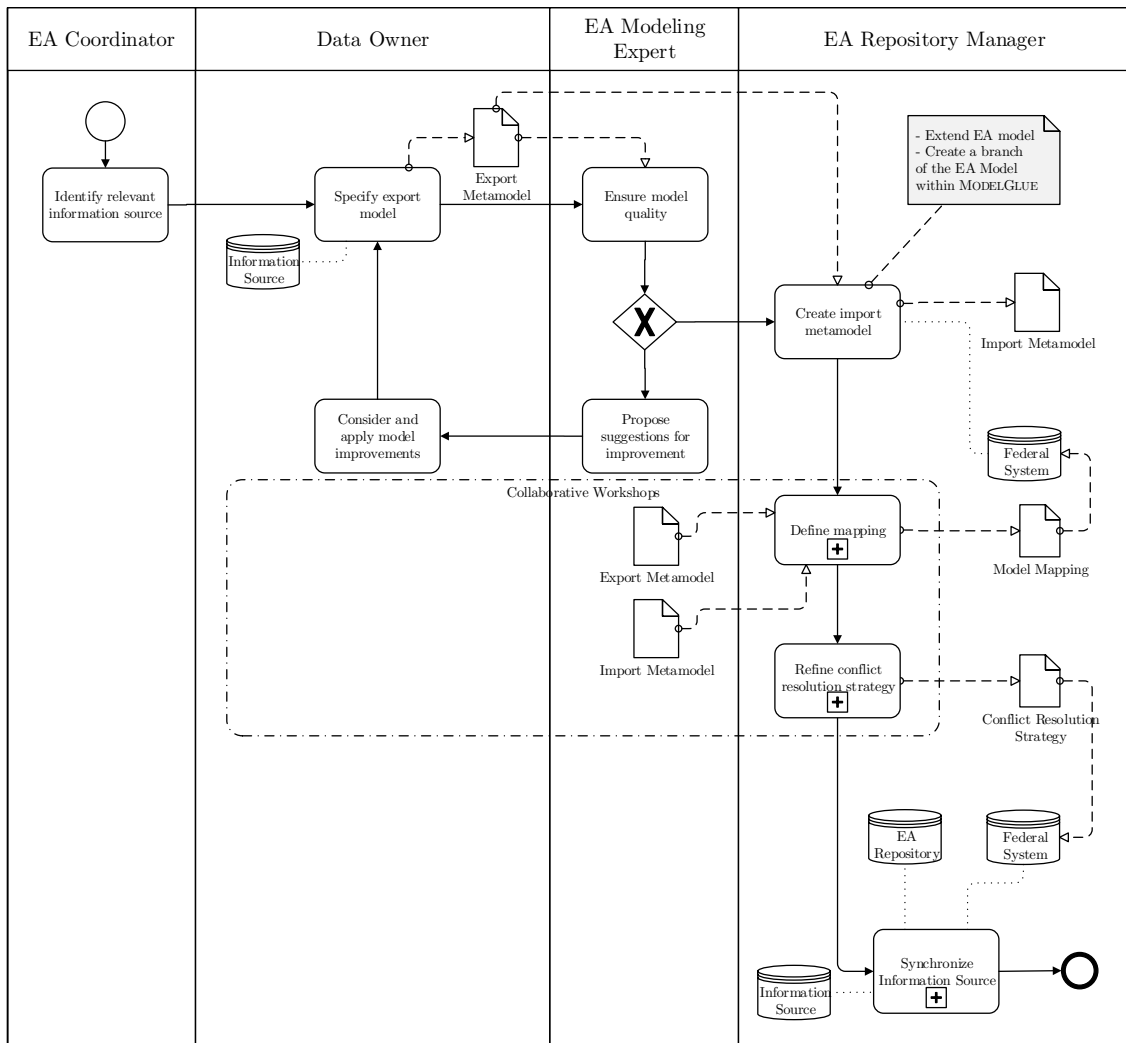


Figure 5.8: Overview of the sub-process integrate information source

legacy systems are always involved in a federated EA model environment and building an IT environment from scratch, i.e. doing a greenfield approach for EA management, is unrealistic. With a greenfield approach, one could use an upper ontology as a frame of reference to infer which information is stored in an information source. We refer to Karfoglou and Schorlemmer [KS03] for a comprehensive overview of the details and Choi et al. [CSH06] for an overview of tools that facilitate the mapping process. Moreover, ontologies are regarded rather counterintuitively when it comes to their definition. Hence, the configuration of a mapping becomes even more time-consuming. Besides these factors, ontology matching itself still faces challenges [SE08].

There are many ways to define a model mapping that go far beyond the scope of the present thesis. The interested reader finds an extensive discussion on the topic by Rahm and Bernstein in [RB01b]. Subsequently, we describe a pragmatic approach that we found useful in the case studies presented in Chapter 7 and other occasions where we

implemented a mapping to an information source, e.g. [Or13]. It embraces an alignment of the different terminology involved, a logical, and a physical mapping of the intended part of an information source to an EA repository.

5.2.2.2 Terminology Alignment

In a first step, the different terms used to describe a concept within information sources must be aligned. This is a highly collaborative effort and is commonly carried out in form of a workshop employing white boards, flip-charts or pen and paper to facilitate communication. Typical attendees of such a workshop meeting are the data owner, the EA coordinator, and the EA modeling expert. Although the goals of this step are

- to find or develop and agree upon a description for a concept,
- to identify similar or equal concepts, and
- to identify possible synonyms, homonyms, hyponyms and hyperonyms,

relationships among concepts including concepts which may be beyond the scope of Federated EA Model Management is very common.

Often it helps to model by example, i.e. talk about concrete instances within an information source instead of the concept behind the instance, cf. [Zl77]. On a larger scale, spreadsheets are used to keep track of the alignment. Example 5.4 elaborates such a spreadsheet-based alignment. Such an initial conceptual alignment is helpful—especially when Federated EA Model Management is executed at a large scale, i.e. with many information sources.



EXAMPLE 5.4: Spreadsheet-based conceptual alignment

The alignment embraces names of classes, i.e. the name of an OBJECTDEFINITION, within the EA model and the model of an information source. Besides different names that refer to the same concept the sheet embraces a description for the core meaning of the concept.



5.2.2.3 Logical Mapping

In the next step, logical mapping takes place. It is developed in a collaborative fashion often carried out in a series of workshops, too. First, the conceptual alignment is perused and concepts are detailed by describing their attributes. Typically, the data owner, EA coordinator, EA repository manager and an EA modeling expert attend such a workshop. At a more mature stage, also developers may attend such a meeting since they can provide detailed knowledge for a specific information source.

The goal during the workshop is to develop a logical mapping that serves

- to identify similar or equal attributes of concepts,
- to align concepts on an attribute level,
- to identify possible abstraction gaps, and
- to describe in which fashion (commonly a database table) the objects and attributes are stored in.

During the meeting the different structures are discussed using OO concepts, i.e. objects, types, attributes, and relationships etc. The terminology alignment is extended by attributes. It depends on the expertise of the participants whether data types are also included or these issues are delayed to the next step, the physical mapping.

Generally, for model or schema mappings, we distinguish [KGI⁺13]:

- **unidirectional** — information is pushed or pulled out of an information system *a* into an information system *b*, and
- **bidirectional** — information is pushed or pulled out of an information system *a* into an information system *b* and if this information is changed in *b*, these changes are propagated (push or pull) to *a*.

This not only holds true for technical mappings but also applies to the logical flow of information in Federated EA Model Management. In this vein, an orthogonal dimension is the kind of interface between the information source and the federal system. Figure 5.9

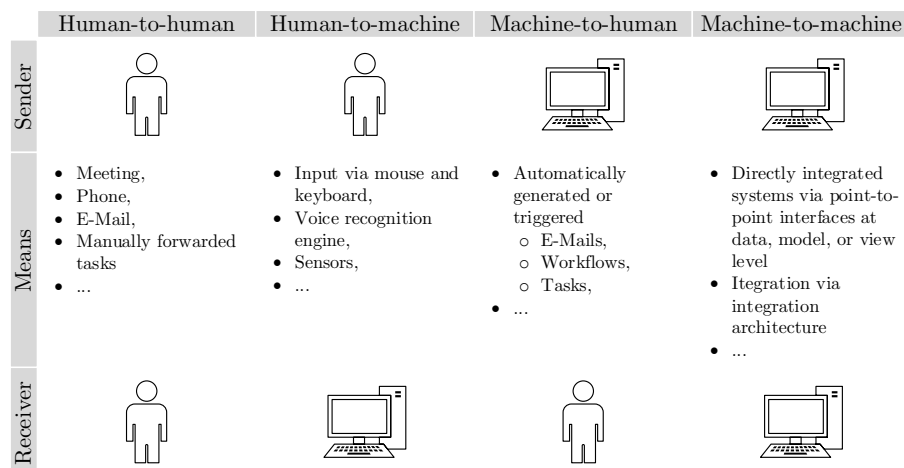


Figure 5.9: Different types of communication within a federated EA model environment

illustrates the different types of communication which can take place between an information source and a federal system 1) to exchange information and 2) to resolve a conflict.

- **Human-to-human** which can be realized through manually forwarding tasks in a workflow engine. Other communication media such as e-mail or phone conferences might be used. In particular complex for issues, workshops are an appropriate means.
- **Human-to-machine** refers to manual user input in an information system commonly done utilizing an advanced UI.
- **Machine-to-human** describes an e-mail or task in an information system that is generated and sent to a user (semi-)automatically.
- **Machine-to-machine** denotes the fully automated communication realized by a technical interface between information systems.

Note that although humans are sender or receiver of information, it is common to store information in an EA repository for further processing. However, means used to communicate a particular piece of information vary considerably. The different types of communication can be combined, i.e. applied as a sequence. For instance, after communicating in a workshop (human-to-human), the information is entered in an information source or the EA repository (human-to-machine). The outcome of this step is an extensive logical mapping of elements within an information source that should be mapped to an EA model.

5.2.2.4 Quality Assessment

In Section 4.1.2, we pointed out that the data quality of different information sources varies considerably. We distinguish between perceived data quality and actual data quality.

Perceived data quality can be measured via: Actual data quality can be measured via:

- | | |
|----------------------------|---|
| • workshops, | • correctness in tables, |
| • face-to-face interviews, | • existence of null values, |
| • phone interviews, | • completeness, i.e. amount of null values, |
| • questionnaires, | • topicality of data, |
| • ... | • ... |

Whether perceived or actual data quality is measured depends on the intended target quality of a specific model element within the EA model. If a low target quality is tolerated, a survey on the perceived quality can be sufficient to accomplish the desired goals. Target quality on the other hand depends on criticality to answer a particular concern (cf. Section 2.1.2). If the element is of high importance to address a concern of EA stakeholders, the target quality must be considerably higher than the target quality of less important elements that describe details.

Figure 5.10 depicts a taxonomy introduced by Rahm and Do in [RD00]. This taxonomy is also discussed by Hinrichs in his PhD thesis [Hi02, p. 29] and Bauer and Günzel [BG13, p. 49]. However, in the context of Federated EA Model Management it provides a classification of problems for information sources divided by single- and multi-source and

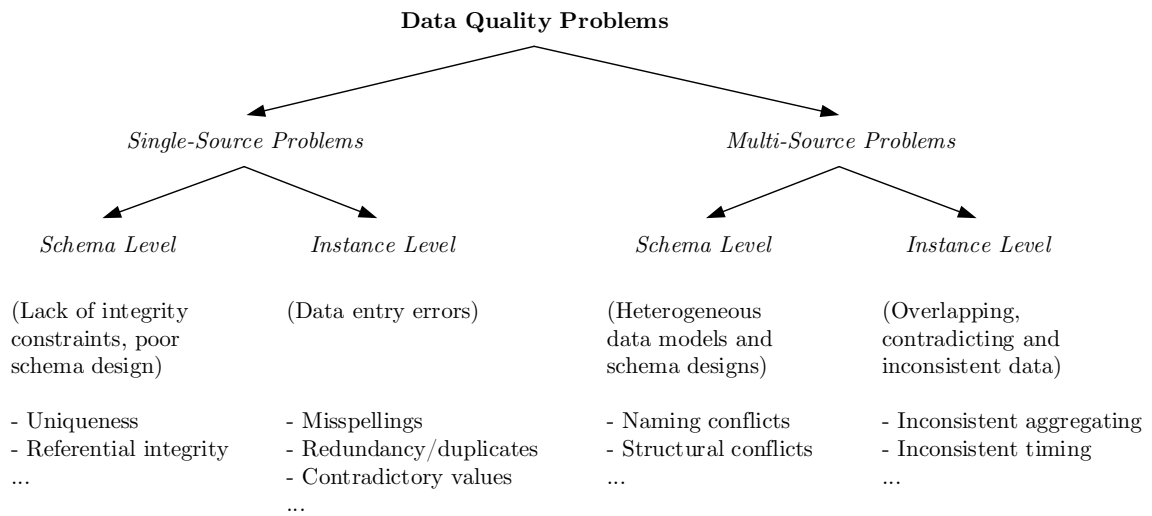


Figure 5.10: A taxonomy of data quality problems according to Rahm and Do [RD00]

schema- and instance-level problems. Rahm et al. further outlined the major steps for data transformation and data cleaning and emphasize the need to cover schema- and instance-related data transformations in an integrated way. Although their research originates in the DBMS area, it is of high relevance for Federated EA Model Management. During case studies and interview series, we found out that the definition of a master-slave relationship (cf. Section 7.5.5 and Definition 5.3) for contradicting information may help to resolve some issues.

DEFINITION 5.3: Master-slave relationship

A master-slave relationship denotes a strategy for *conflict avoidance* in Federated EA Model Management. Let us assume $\mathcal{E}_a, \mathcal{E}_b$ are boundary model elements with respective information sources $\mathcal{I}_a, \mathcal{I}_b$. We call \mathcal{I}_a *master* and \mathcal{I}_b is *slave* if changes on \mathcal{E}_a are always preferred over changes on \mathcal{E}_b in the course of a model merge. ■

For an extensive discussion on data quality, we refer the interested reader to Batini et al. [BCF⁺09]. The authors provide a comprehensive overview of relevant literature which covers a wide range of approaches to not only assess but also to improve the quality of information.

5.2.2.5 Physical Mapping

The terminology alignment and logical mapping serve as a starting point for the development of a physical mapping. Although attributes and physical data types already may be documented, in this step, these technical details become very important. They include but are not limited to [BG13, p. 57] (cf. also Section 4.2.3 and Section 5.2.8.2):

- transforming and harmonizing data types,

- converting between different encodings,
- harmonization of strings,
- harmonization of dates,
- conversion of different units, and
- combining or splitting values.

In order to develop a physical mapping, a concrete Generic Data Model (GDM) must be used that serves as a common denominator among the different information sources (cf. Figure 4.9). This GDM can be considered the target metamodel. The target metamodel of MODELGLUE is a subset of the metamodel presented in Figure 5.2. It is a model with objects, attributes, and relationships between the objects.

In order to apply this model as a target metamodel, one must create the (ontologic) metamodel as a linguistic instance of this metamodel. In the course of an evolution of an information source, this metamodel must be maintained as well. The creation and maintenance of this model can be done either manually or automatically. Manual maintenance of the metamodel is commonly required for the retrieval and import of information from a Relational Database (RDB) or from a spreadsheet. Alternatively, automated approaches can be considered for information systems which feature metamodeling facilities, such that the different metamodels can be mapped.

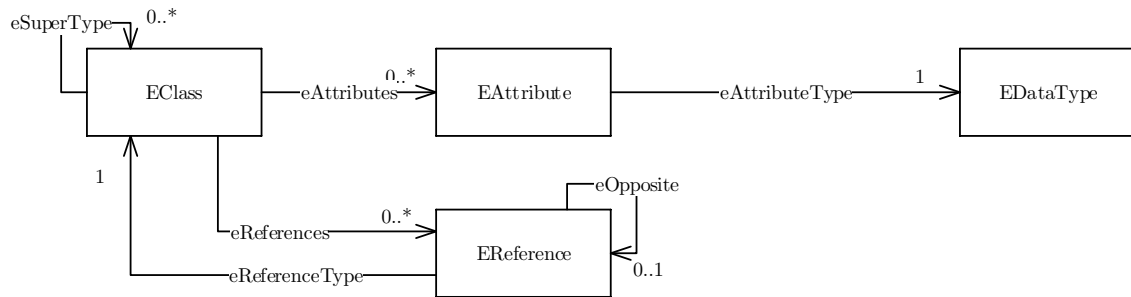
In practice, a multitude of metamodels exist. In the following, we illustrate one example, namely Ecore as part of the EMF project. Ecore can be used for an automated model as well as metamodel synchronization realized through a mapping of the metamodels of Ecore with MODELGLUE. After explaining this mapping in more detail, we elaborate two additional examples, namely RDBs and spreadsheets. These are often used as interface techniques in situations, in which the manual creation and adaptation of a metamodel is common (cf. Chapter 7).

5.2.2.5.1 Mapping Ecore to MODELGLUE

Ecore [TEF14] is a very prominent example of an open-source platform that offers metamodeling facilities. Its core entities and their relationships are depicted in Figure 5.11.

The diagram consists of the following entities [SBP⁺09, ch. 5]:

- **EClass** models classes which are identified by a name. They can have a number of attributes and references. Inheritance is realized by a reference to other classes which are considered supertypes. In MODELGLUE, EClasses map to OBJECTDEFINITIONS.
- **EAttribute** models attributes which are identified by their name, too. Additionally, EAttributes have a type (see EDataType). In MODELGLUE, EAttributes map to ATTRIBUTEDEFINITIONS.

Figure 5.11: Simplified view of the Ecore kernel based on [SBP⁺09, ch. 5]

- **EReference** is used to model associations between classes. An EReference models one end of such an association. Like attributes and classes, references are identified by name. Similar to attributes, references have a type. In contrast to attributes, however, this type must be the EClass at the other end of the association. Similar to Attributes within MODELGLUE, an association that is navigable in the opposite direction is expressed by another EReference to denote this bidirectional relation. Each EReference further specifies lower and upper bounds to express cardinalities. In MODELGLUE, EReferences map to ATTRIBUTEDEFINITIONS.
- **EDataType** denotes the primitive data types. These EDataType map to data types defined in Java, e.g. String, int, etc. EDataType are used to specify a type of an attribute. In MODELGLUE, EDataTypes map to TYPES whereas the notion of TYPES is more volatile than an EDataTypes. Commonly, TYPES in MODELGLUE are interpreted when accessing VALUES.

To illustrate a mapping of both metamodels, we need to examine an additional concept of EMF, namely an instance of an EClass which is called *EObject*. Since an introduction of the exact inheritance and relations between the different concepts in EMF is beyond the purpose of the present thesis, we refer the interested reader to [SBP⁺09, ch. 5] and continue under the assumption that an EOBJECT is capable to store EATTRIBUTES and EREFERENCES as its contents.

Table 5.1(a) depicts the mapping of the metamodels of Ecore and MODELGLUE. This mapping builds the foundation for fully automated, bidirectional exchange of metamodels which are based on Ecore or MODELGLUE. However, as we discussed above within Federated EA Model Management unidirectional approaches prevail; especially the metamodel is commonly synchronized unidirectionally.

Table 5.1(b) shows the mappings on instance level to exchange information stored in models. We assume that an EOBJECT stores EATTRIBUTES and EREFERENCES as its content. Both mappings reflect that MODELGLUE treats relationships as ‘ordinary’ attributes.

5.2.2.5.2 Mapping Spreadsheets to MODELGLUE

A frequently offered export format of commercial tools is Comma-Separated Values (CSV) or Microsoft Excel Spreadsheet Format (XLS/XLSX). Although these formats are well-known

ECore 2.0	MODELGLUE
EClass	ObjectDefinition
EAttribute	AttributeDefinition
EReference	AttributeDefinition
EDataType	Type

(a) Schema mapping

ECore 2.0	MODELGLUE
EObject	Object
EAttribute	Attribute
EAttribute	Type
EAttribute	Value
EReference	Attribute
EReference	Type
EAttribute	Value

(b) Instance mapping

Table 5.1: Mapping of the Ecore kernel and the core concepts of MODELGLUE

and relatively simple, the exact mapping of information to rows, columns, and sheets varies considerably. However, during our case-studies and during occasional imports in more than four years, we experienced a certain consolidation in the number of approaches to export spreadsheets taken by metamodeling platforms. In the following, we sketch a pattern that we observed, i.e.

- One sheet per type, i.e. for each type a separate sheet is used
- One row per object, i.e. for each object a separate row is used
- One column per attribute, i.e. for each attribute a separate column is used

This general structure sketches a solution space that varies considerably. Hence, we outline the variants observed, i.e.

- Using a unique identifier for each object
- Relationships can be defined via URIs, referenced cells in the same or a different sheet, or (exact) name matching
- Multi-valued attributes can be exported via a multi-line cell with a delimiter, e.g. the newline character, or for n-valued attributes one can use n-columns

5.2.2.5.3 Mapping Relational Databases to MODELGLUE

Figure 5.12 depicts the target metamodel of MODELGLUE which represents a subset of the metamodel presented in Figure 5.2 and a mapping to database tables. We assume that each type a real-world object is modeled as a separate table, columns are used to model attributes as well as foreign key columns to model relationships with cardinality 1..n and n..1 respectively. Moreover, relationship tables are used to model n..m relations. Each real-world object is then stored as a single row of a table that is identified by a unique primary key.

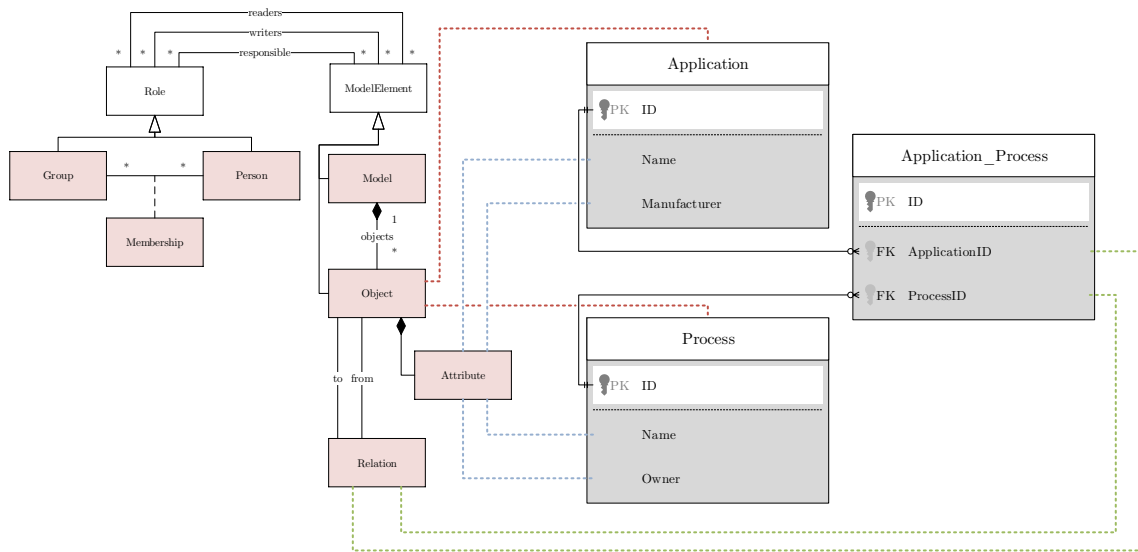


Figure 5.12: Illustrative mapping of a relational database to the target metamodel

In [Or13], Orhan utilizes Extensible Markup Language (XML) files to specify mappings between RDBMS and models which conform to the structure illustrated in Figure 5.12. It shows how database tables can be related to this import model.

To create XML-based mappings, sophisticated tools, e.g. [A114], exist that allow users to define mappings graphically. We refer the interested reader to Kirschner [Ki14] and Orhan [Or13]; in their master's theses, both authors develop mappings for the outlined target structure and import real-world data. The core structure as well as the entire XML-Schema to develop such a mapping is provided in Appendix A. Besides the specification of model-to-model transformations via Structured Query Language (SQL) queries it allows to define access-rights for each model element that is imported. These can also be imported or generated based in information stored within the RDBMSs using the SQL-queries, i.e. each mapping commonly can incorporate vendor-specific SQL-queries.

5.2.3 Importing Information

Figure 5.13 shows the flow between MODELGLUE, which realizes a basic scheduling for the different information sources, its importer component, and an information source. The importer writes information to the import model whereas the information source pulls information from the export model. Since the importer pulls information from the source, the information source is not modified (cf. Assumption 4.2). If an object identified with its OID already exists within the import model, a new version is created that captures current information provided by the export model.

Both alternatives in Figure 5.13 result in an object that is finally returned. Besides meta-information and the object name, attribute changes and changes of links between objects have to be imported. Figure 5.14 illustrates schematically how an object's attributes are updated with information of a source. In a first step, each attribute is either created or

5. Federated EA Model Management Design

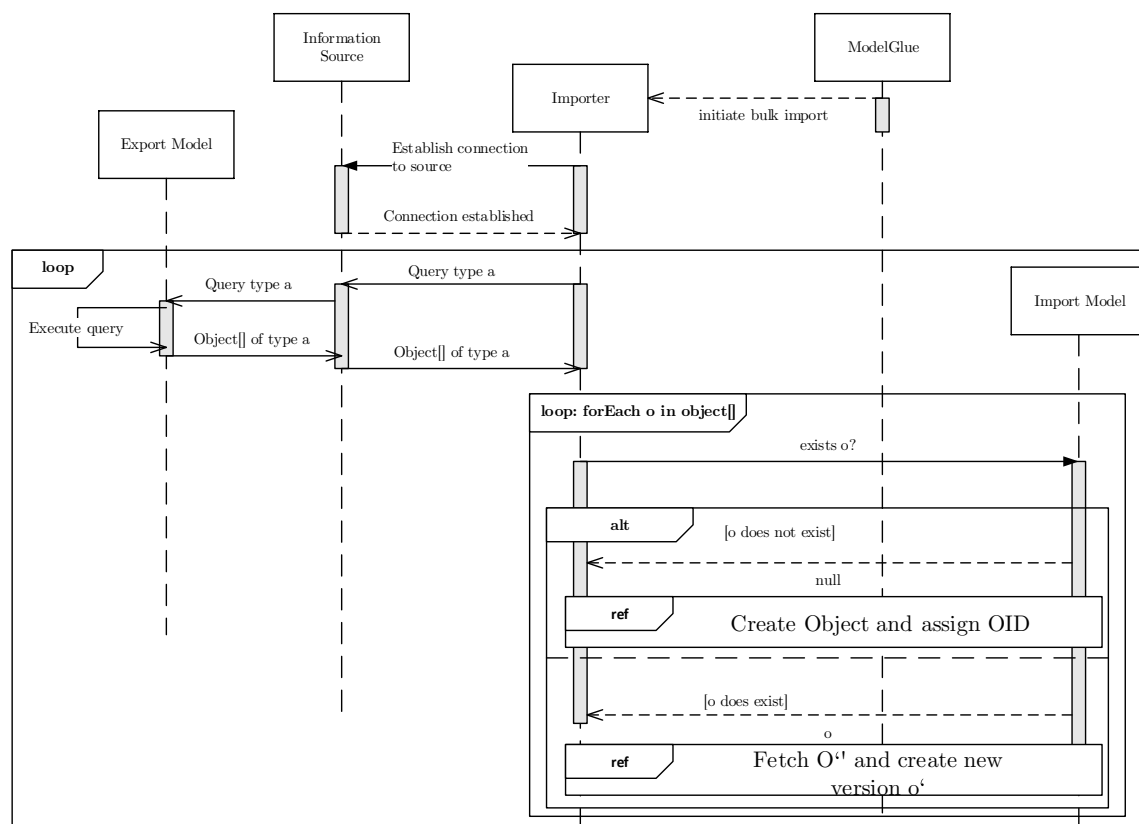


Figure 5.13: Importer: initiating a connection and querying for objects

updated if changes occur, i.e. this schematic figure depicts an update scenario where we rely on the OID to find the provenance. We further must distinguish single-valued attributes and multi-valued attributes. In addition to the sketched updates, the update of links is an important step. In this context, it might be relevant to create object stubs within the system. One could either ensure to create all objects first and then update their attributes or use stubs as a target link. Objects within the import model that are not contained within the information source must be removed similar to missing attributes. In particular when no OID exists, this approach is considerable error-prone when one must rely on other techniques such as exact name matching.

In line with the different mechanisms to get information, which are described in Section 4.2.5, an Application Programming Interface (API) could also return a log file of the performed operations³. In such a case, one has to apply these logged changes to the import model. This could require a model-to-model transformation of the query language used within the log file to the query language of the information system which contains the import model. Although an eventually required transformation of query languages is necessary, this mechanism provides clear advantages. In contrast to the sketched approach, directly replaying changes minimizes not only the amount of transferred information but also does

³A prominent example of an API that is widely used is OData [ODa13]. One could use OData operations to represent applied changes in an information system.

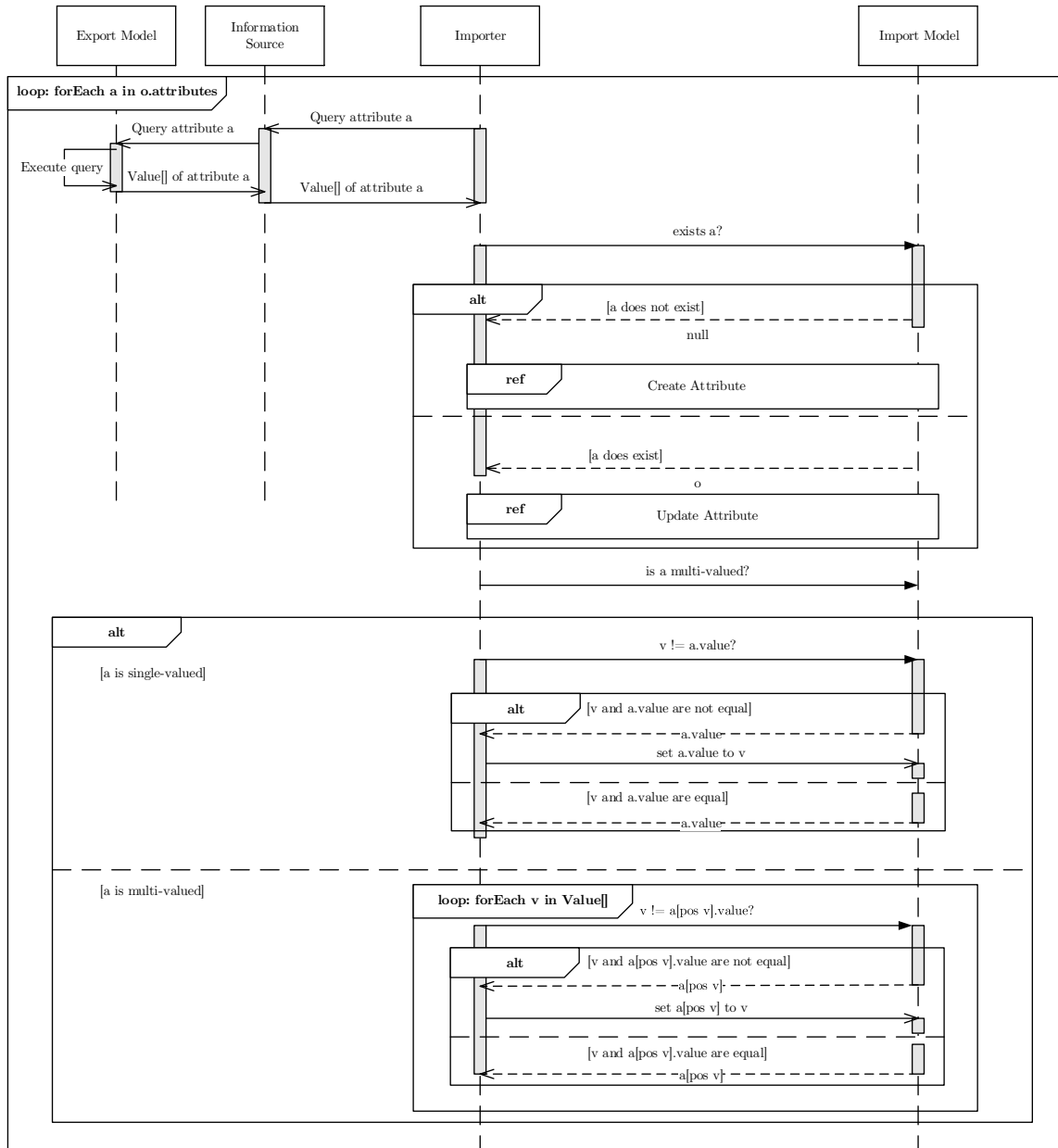


Figure 5.14: Importer: updating an object

not require to traverse the entire import model. Another advantage of this approach is given by the ability to trace updates on object names in the import model if no OID exists within the source system. If one has to rely on exact name matching, an updated object appears as newly created object—a change that cannot be detected and reapplied. Instead, the original object and its entire history is deleted and a new object is created. While information of performed operations is desirable, often fine-grained information on performed operations is not available within an information source.

Whether dumps must be compared or operations can be replayed, in the course of an import, the physical mapping is executed. As already mentioned in Section 4.2.5, an import can be triggered manually, time-based, or triggered in an automated fashion based on certain events, see e.g. [FSB⁺12].

5.2.3.1 Metamodel Changes

Changes within a metamodel are most likely to occur in the course of a change request within an information source. As we depicted in Section 6.2, the exchange of a model and metamodel could be capabilities of an information source. However, this is not always the case. In turn, this means if the information source is not capable to inform the federal system about changes within its metamodel, a manual trigger and reconfiguration of the mapping (cf. UC2) has to be integrated into the change management process of an organization.

5.2.3.2 Important parameters

Even in cloud environments, resources are finite. Commonly system load rises linearly until resource pools are exhausted (cf. [FGP⁺08, p. 250]). Then load rises non-linearly and commonly can be described with a polynomial function. Response behavior is easy to predict when system load rises linearly. In contrast, it is hard to predict when it rises in a polynomial manner since caches, paging/swapping, queues, etc. are utilized to overcome limited resources. Respective polynomial functions that describe the load behavior vary considerably and depend on the actual load.

In this vein, the chunk size and timing specificities with regard to an automated periodical synchronization of information are of high relevance. During production, a bulk import of all information might not be possible since this would influence the information source's runtime behavior. Specifying a chunk size limits the size of information requested to a reasonable amount such that the information can be synchronized incrementally.

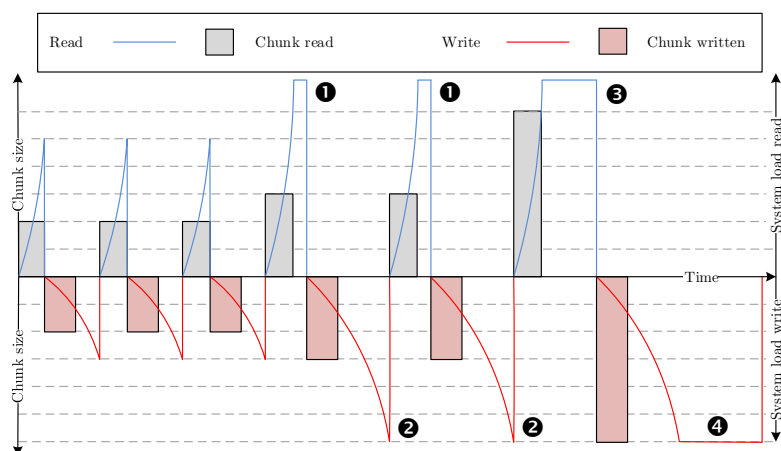


Figure 5.15: Relationship between chunk size and system load (schematic)

Figure 5.15 illustrates this relationship more clearly. As the system reads or writes more data at once, its load rises. If resources are exhausted during read (❶ in Figure 5.15), the time rises non-linearly—possibly exponentially. The graphic also illustrates that reading information is commonly faster than writing it to a persistent storage. This behavior strongly depends on the underlying hardware but generally holds true for magnetic drives as well as solid-state drives. Even if the import model can write larger chunks of data in almost linear time (❷ in Figure 5.15), there is an upper bound which—in the course of importing information—causes the systems of the export model as well as the import model (❸ and ❹ in Figure 5.15) to run into bottlenecks. Again, during these bottlenecks (❶, ❷, ❸, ❹ in Figure 5.15) system response cannot be predicted adequately. While Figure 5.15 describes the relation of one information source’s load to the import model, in Federated EA Model Management, multiple information sources must be considered such that the parameters

- **chunk size** of information imported at once,
- **period** how frequently the information is imported, and
- **cache size** of the import model

as well as the involved hardware resources must be carefully balanced.

In Figure 5.15, we assume that load commonly rises linearly till no further resources, such as memory and Central Processing Unit (CPU) time, are available; then it rises exponentially and reaches its peak. When this peak is reached, the reaction of the system could be influenced. This may lead to unresponsive behavior or even to system failures. Consequently, synchronizing an information source during operations should take place using a chunk size that does not impact the system substantially. In contrast, a synchronization process executed during times when the information source is not used for operations can use a larger chunk size. Note that Figure 5.15 abstracts from concrete load profiles and completely omits any latencies that are typical for distributed systems and load produced due to productive day-to-day usage. This illustration is meant to outline the schematics of the causal relationship between time intervals, the chunks read and written, and system load produced rather than an exact model for system load behavior.

The goal during an import is not to reach any bottlenecks while information must be exchanged in a timely manner. Especially time constraints depend on the intent of the model element of the information source synchronized with the EA model. Another important aspect on the writing end, i.e. the import model, is the cache size used for objects and attributes within MODELGLUE. Generally speaking, the larger this cache size is, the better the performance behavior. However, caches require resources, i.e. memory, and, thus, it must be sized for each information source individually as well.

To find the right chunk size and other important factors, empirical data is needed, which can be gathered using profiling techniques to measure load behavior. We refer the interested reader to Ford et al. [FGP⁺08, p. 245ff]. The authors elaborate on capacity planning, provide best practices, and illustrative examples.

5.2.4 Model Differencing

To ensure model quality, we provide means to compare two models in a visual fashion.

DEFINITION 5.4: Model differencing

Model differencing describes the calculation process of deviations between two model elements. ■

Brun and Pierantonio refer to the problem of model differencing as intrinsically complex [BP08]. They propose a coarse-grained structure that helps to analyze and understand diverse differencing approaches. Brun and Pierantonio divide the problem in

- **calculation**, i.e. an algorithm able to compare models,
- **representation**, i.e. a model that is used to represent the outcome of the algorithm, and
- **visualization**, i.e. a human-readable format for differences easy to grasp for stakeholders.

In the remainder of this section, we report on the calculation (Section 5.2.4.2) and representation (Section 5.2.4.1) whereas the visual aspect of model differences is covered in Section 5.3.1.

In the algorithm design, we combine concepts of two-way and three-way state-based differencing (cf. Section 2.3.1), i.e. the algorithm compares two models since many users are familiar with visualizations of comparisons between two objects (e.g. source code). We aim at an intuitive understanding of visual concepts; thus, differencing is restricted to pairwise comparisons (two-way). However, if the model differencing algorithm detects differences, the base revision is used for showing a three way comparison.

In Figure 5.16, we show the different models involved in the algorithm and their intersection. The general idea of the difference algorithm is to compare `OBJECTDEFINITIONS` that are in $\mathcal{M}_a \cap \mathcal{M}_b$. Whereby $\mathcal{M}_a \cap \mathcal{M}_b$ is calculated using the congruence operator (cf. ‘ \cong ’ in Section 5.2.4.3). However, for our purposes we transform all `MODELELEMENTS` in $\mathcal{M}_a \Delta \mathcal{M}_a$ to placeholders which conform to the differencing model. These ‘stubs’ carry no differences and serve for later display in the differencing visualization (cf. Section 5.3.1). However, they conform to a certain metamodel. This metamodel can be processed by the visualization algorithm and is introduced in the following.

5.2.4.1 A Metamodel for Model Differencing

Figure 5.17 illustrates the metamodel to calculate the differences between different models, their `OBJECTS`, `ATTRIBUTES` and respective `DEFINITIONS`. It constitutes a subset of the metamodel introduced in Section 5.1.2. In addition to the concepts explained in Section 5.1.2, the metamodel illustrated in Figure 5.17 incorporates a concept called `DIFFERENCE` which saves information about two versions a, b and the *origin* of a `MODELELEMENT`.

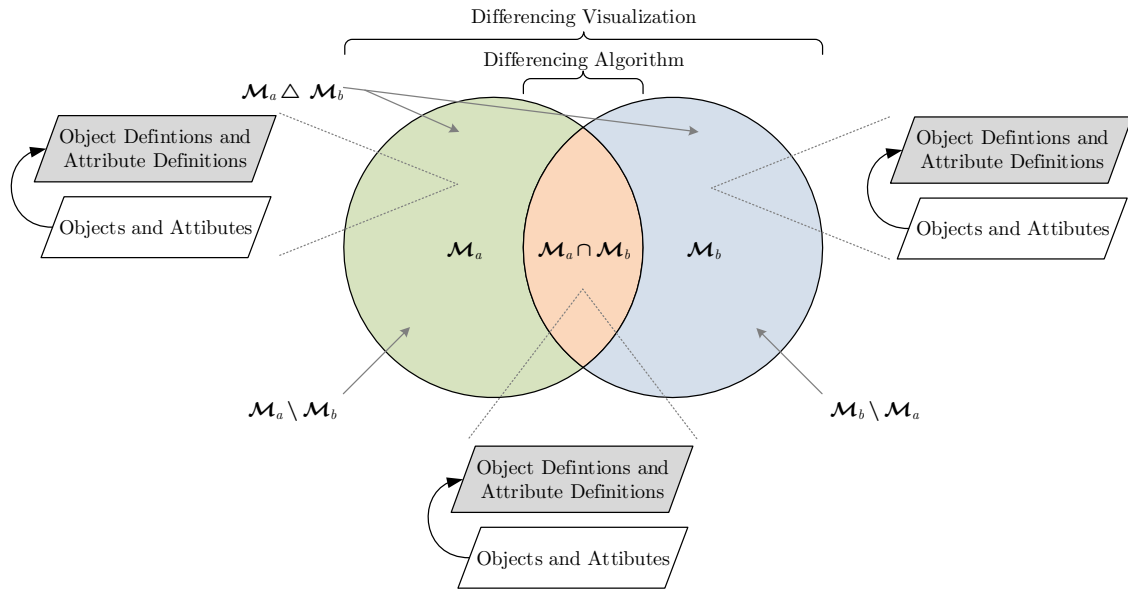


Figure 5.16: Scope of the differencing algorithm and the differencing visualization as an annotated Venn diagram

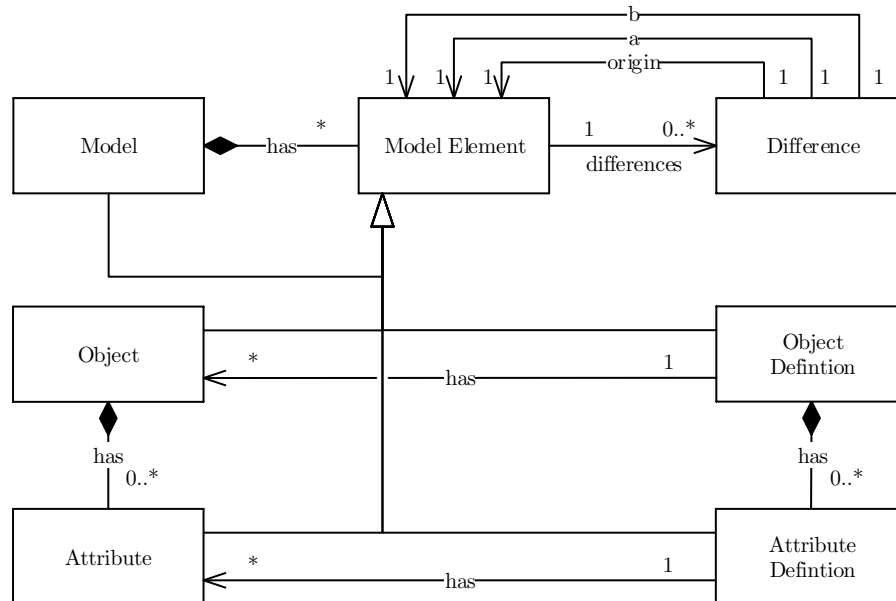


Figure 5.17: A metamodel for calculating differences between different models

5.2.4.2 An Algorithm for Model Differencing

Algorithm 3 compares two models with each other. This is accomplished by looking at a particular state of each model, i.e. the latest revision of two branches. The algorithm takes two models \mathcal{M}_a , \mathcal{M}_b that conform to the metamodel depicted in Figure 5.2 and returns

the differences of these models as an instance of the metamodel shown in Figure 5.17. This model can be used as an information base for visualizing the differences (cf. Section 5.3.1).

Function compareModels (line 1) takes two MODELS that conform to the metamodel as introduced in Section 5.1.2 and calculates model as well as metamodel differences.

Line 2: In a first step, an empty model of the structure illustrated in Figure 5.17 is initialized which carries the results.

Line 3–4: In a second step, all intersecting OBJECTDEFINITIONS are analyzed.

Line 5–8: For all other OBJECTDEFINITIONS, placeholders are generated which later on serve to visualize not only elements that carry differences but also to visualize the remaining MODELELEMENTS.

Line 9–10: In a final step, all OBJECTS are investigated for differences. Found differences are stored in the globally available model \mathcal{M}_c which is returned by the algorithm.

Function compareObjectDefinitions (line 11) compares two OBJECTDEFINITIONS with each other.

Line 12: Differences are calculated by invoking a generic differencing method. Differences are then passed to a new OBJECTDEFINITION that conforms to the metamodel depicted in Figure 5.17.

Line 13: The ATTRIBUTEDEFINITIONS of the currently investigated OBJECTDEFINITIONS are analyzed for differences and added to the newly created OBJECTDEFINITION.

Line 14: The result of the differencing of the two OBJECTDEFINITIONS and their ATTRIBUTEDEFINITIONS is finally added to the model.

Function compareObjects (line 15) takes two OBJECTS as input and calculates their differences.

Line 16: A new OBJECT that conforms to the metamodel depicted in Figure 5.17 is created. Thereby, the results of a difference calculation is passed to the newly created OBJECT.

Line 17: The newly created differencing OBJECT is part of the produced result of the algorithm. As such, it is added to the model.

Line 18: All ATTRIBUTES of the currently investigated OBJECTS are analyzed for differences. Note that relationships are also included as they are represented as ordinary ATTRIBUTES.

Line 19–20: The object tree is traversed recursively in case any of the involved OBJECTS exhibits children. Note that the same congruence operator (cf. Section 5.2.4.3) is used to determine related children; only these are traversed.

Line 21–24: For all non-related objects the same argument is passed as first and second parameter for the differencing such that, effectively, the result serves for visualization purposes only and does not carry actual differences.

Function compareAttributes (line 25) compares all attributes of two given OBJECTS with each other returning their differences.

Algorithm 3: Differencing of models

Input: Model $\mathcal{M}_a, \mathcal{M}_b$, Baseline Time t_b
Output: Model \mathcal{M}_c

```

1: function compareModels( $\mathcal{M}_a, \mathcal{M}_b$ ):
2:    $\mathcal{M}_c \leftarrow \{\emptyset\}$ 
3:   foreach  $\mathcal{D}_a^\mathcal{O} \in \mathcal{M}_a.objectDefinitions, \mathcal{D}_b^\mathcal{O} \in \mathcal{M}_b.objectDefinitions$  ( $\mathcal{D}_a^\mathcal{O} \cong \mathcal{D}_b^\mathcal{O}$ ) do
4:      $\lfloor$  compareObjectDefinitions( $\mathcal{D}_a^\mathcal{O}, \mathcal{D}_b^\mathcal{O}$ )
5:   foreach  $\mathcal{D}_a^\mathcal{O} \in (\mathcal{M}_a.objectDefinitions \setminus \mathcal{M}_b.objectDefinitions)$  do
6:      $\lfloor$  compareObjectDefinitions( $\mathcal{D}_a^\mathcal{O}, \emptyset$ )
7:   foreach  $\mathcal{D}_b^\mathcal{O} \in (\mathcal{M}_b.objectDefinitions \setminus \mathcal{M}_a.objectDefinitions)$  do
8:      $\lfloor$  compareObjectDefinitions( $\emptyset, \mathcal{D}_b^\mathcal{O}$ )
9:   compareObjects( $\mathcal{M}_a.root, \mathcal{M}_b.root$ )
10:  return  $\mathcal{M}_c$ 

11: function compareObjectDefinitions( $\mathcal{D}_a^\mathcal{O}, \mathcal{D}_b^\mathcal{O}$ ):
12:   $\mathcal{D}_{\Delta(a,b)}^\mathcal{O} \leftarrow$  new OBJECTDEFINITION(createDifference( $\mathcal{D}_a^\mathcal{O}, \mathcal{D}_b^\mathcal{O}$ ))
13:  compareAttributeDefinitions( $\mathcal{D}_a^\mathcal{O}, \mathcal{D}_b^\mathcal{O}, \mathcal{O}_{\Delta(a,b)}$ ) /* similar to compareAttributes(...) */
14:   $\mathcal{M}_c.objectDefinitions \leftarrow \mathcal{M}_c.objectDefinitions \cup \mathcal{D}_{\Delta(a,b)}^\mathcal{O}$ 

15: function compareObjects( $\mathcal{O}_a, \mathcal{O}_b$ ):
16:   $\mathcal{O}_{\Delta(a,b)} \leftarrow$  new OBJECT(createDifference( $\mathcal{O}_a, \mathcal{O}_b$ ))
17:   $\mathcal{M}_c.objects \leftarrow \mathcal{M}_c.objects \cup \mathcal{O}_{\Delta(a,b)}$ 
18:  compareAttributes( $\mathcal{O}_a, \mathcal{O}_b, \mathcal{O}_{\Delta(a,b)}$ )
19:  foreach  $child_a \in \mathcal{O}_a.children, child_b \in \mathcal{O}_b.children$  ( $child_a \cong child_b$ ) do
20:     $\lfloor$  compareObjects( $child_a, child_b$ ) /* traverses the OBJECT tree recursively */
21:  foreach  $child_a \in (\mathcal{O}_a.children \setminus \mathcal{O}_b.children)$  do
22:     $\lfloor$  compareObjects( $child_a, \emptyset$ ) /* traverses the OBJECT tree recursively */
23:  foreach  $child_b \in (\mathcal{O}_b.children \setminus \mathcal{O}_a.children)$  do
24:     $\lfloor$  compareObjects( $\emptyset, child_b$ ) /* traverses the OBJECT tree recursively */

25: function compareAttributes( $\mathcal{O}_a, \mathcal{O}_b, \mathcal{O}_{\Delta(a,b)}$ ):
26:  foreach  $\mathcal{A}_a \in \mathcal{O}_a.attributes, \mathcal{A}_b \in \mathcal{O}_b.attributes$  ( $\mathcal{A}_a \cong \mathcal{A}_b$ ) do
27:     $\lfloor$   $\mathcal{O}_{\Delta(a,b)}.attributes \leftarrow \mathcal{O}_{\Delta(a,b)}.attributes \cup$  new ATTRIBUTE(createDifference( $\mathcal{A}_a, \mathcal{A}_b$ ))
28:  foreach  $\mathcal{A}_a \in (\mathcal{O}_a.attributes \setminus \mathcal{O}_b.attributes)$  do
29:     $\lfloor$   $\mathcal{O}_{\Delta(a,b)}.attributes \leftarrow \mathcal{O}_{\Delta(a,b)}.attributes \cup$  new ATTRIBUTE(createDifference( $\mathcal{A}_a, \emptyset$ ))
30:  foreach  $\mathcal{A}_b \in (\mathcal{O}_b.attributes \setminus \mathcal{O}_a.attributes)$  do
31:     $\lfloor$   $\mathcal{O}_{\Delta(a,b)}.attributes \leftarrow \mathcal{O}_{\Delta(a,b)}.attributes \cup$  new ATTRIBUTE(createDifference( $\emptyset, \mathcal{A}_b$ ))

32: function createDifference( $\mathcal{E}_a, \mathcal{E}_b$ ):
33:  if  $\mathcal{E}_a \equiv \mathcal{E}_b$  then
34:     $\lfloor$  return new DIFFERENCE( $\mathcal{E}_a, \mathcal{E}_b, EQUAL$ )
35:   $\mathcal{E}_{base} \leftarrow$  getBaseVersion( $\mathcal{E}_a, t_b$ )
36:  if  $\mathcal{E}_{base} \equiv \emptyset \wedge \mathcal{E}_a \equiv \emptyset$  then
37:     $\lfloor$  return new DIFFERENCE(NEW,  $\mathcal{E}_b$ , NEW)
38:  else if  $\mathcal{E}_{base} \equiv \emptyset \wedge \mathcal{E}_b \equiv \emptyset$  then
39:     $\lfloor$  return new DIFFERENCE( $\mathcal{E}_a$ , NEW, NEW)
40:  else if  $\mathcal{E}_{base} \neq \emptyset \wedge \mathcal{E}_a \equiv \emptyset$  then
41:     $\lfloor$  return new DIFFERENCE(DELETED,  $\mathcal{E}_b, \mathcal{E}_{base}$ )
42:  else if  $\mathcal{E}_{base} \neq \emptyset \wedge \mathcal{E}_b \equiv \emptyset$  then
43:     $\lfloor$  return new DIFFERENCE( $\mathcal{E}_a$ , DELETED,  $\mathcal{E}_{base}$ )
44:  else
45:     $\lfloor$  return new DIFFERENCE( $\mathcal{E}_a, \mathcal{E}_b, \mathcal{E}_{base}$ )

```

Line 26–27: The same congruence operator is used to find related attributes, calculate their differences, and create an `ATTRIBUTE` that receives the result of the differencing as an argument. Thereafter, the newly created `ATTRIBUTE` is attached to the `OBJECT`.

Line 28–31: All non-related `ATTRIBUTES` are used to create placeholders that serve visualization purposes only.

Function createDifferences (line 32) Checks for similarity of `MODELELEMENTS` and creates a three-way difference if the `MODELELEMENTS` passed to the function are different.

Line 33–34: The algorithm compares two model elements and returns immediately if they turn out to be equal.

Line 35: The base version for a given time t_b is retrieved. This base version of a `MODELELEMENT` is used for the three-way comparison. The base version, helps to distinguish whether new model element have been created or deleted.

Line 36–39: The cases are handled in which a change is only performed in one of the branches and the model element does not exist in the other branch.

Line 40–43: Deleted model elements are dealt with. Possibly a model element has been deleted in one of the branches and modified in the other branch.

Line 45 The actual differences are stored if all other cases did not match.

5.2.4.3 Equivalence of Model Elements

For the sake of clarity, we abstract from two specificities in Algorithm 3. First, we did not detail how to determine whether two elements are in the same set (\cong). Second, the equivalence (\equiv) of a `MODELELEMENT` in line 33 is determined using different parameters.

In his master’s thesis, Kirschner [Ki14, p. 39] uses the congruence symbol (\cong) for two model elements $\mathcal{E}_1, \mathcal{E}_2$ to denote that either one element is a branch of the other or both elements share a common origin (cf. Equation 5.9).

$$\begin{aligned} \mathcal{E}_1 \cong \mathcal{E}_2 &\Leftrightarrow \mathcal{E}_1.uid \equiv \mathcal{E}_2.oid \vee \\ &\mathcal{E}_1.oid \equiv \mathcal{E}_2.uid \vee \\ &\mathcal{E}_1.oid \equiv \mathcal{E}_2.oid \end{aligned} \tag{5.9}$$

Since OIDs are considered to be URIs, their equivalence can be determined by normalizing as specified in [BLFM05, DS05] and by comparing them for equivalence as described in [FGM⁺99]. This notion of congruence between `MODELELEMENTS` is used to detail the equivalence of the different kinds of `MODELELEMENTS` as used in Algorithm 3.

Let ι be a utility function determining the index within an ordered list of `MODELELEMENTS`, then Equation 5.10 illustrates equivalence between two `ATTRIBUTES` $\mathcal{A}_1, \mathcal{A}_2$.

$$\begin{aligned} \mathcal{A}_1 \equiv \mathcal{A}_2 &\Leftrightarrow \mathcal{A}_1 \cong \mathcal{A}_2 \wedge \\ &\mathcal{A}_1.name = \mathcal{A}_2.name \wedge \\ &\forall \mathcal{V}_1 \in \mathcal{A}_1.values \exists \mathcal{V}_2 \in \mathcal{A}_2.attributes (\mathcal{V}_1 \equiv \mathcal{V}_2 \wedge \iota(\mathcal{V}_1) = \iota(\mathcal{V}_2)) \end{aligned} \tag{5.10}$$

Equation 5.11 illustrates equivalence between two `ATTRIBUTEDEFINITIONS` $\mathcal{D}_1^A, \mathcal{D}_2^A$.

$$\begin{aligned}
 \mathcal{D}_1^A \equiv \mathcal{D}_2^A &\Leftrightarrow \mathcal{D}_1^A \cong \mathcal{D}_2^A \wedge \\
 &\mathcal{D}_1^A.name = \mathcal{D}_2^A.name \wedge \\
 &\forall \mathcal{C}_1 \in \mathcal{D}_1^A.constraint \exists \mathcal{C}_2 \in \mathcal{D}_2^A.constraint (\mathcal{C}_1 \equiv \mathcal{C}_2) \\
 &\forall \mathcal{C}_2 \in \mathcal{D}_2^A.constraint \exists \mathcal{C}_1 \in \mathcal{D}_1^A.constraint (\mathcal{C}_2 \equiv \mathcal{C}_1)
 \end{aligned} \tag{5.11}$$

Equation 5.12 illustrates equivalence between two `OBJECTS` $\mathcal{O}_1, \mathcal{O}_2$.

$$\begin{aligned}
 \mathcal{O}_1 \equiv \mathcal{O}_2 &\Leftrightarrow \mathcal{O}_1 \cong \mathcal{O}_2 \wedge \\
 &\mathcal{O}_1.name = \mathcal{O}_2.name \wedge \\
 &\forall \mathcal{A}_1 \in \mathcal{O}_1.attributes \\
 &\exists \mathcal{A}_2 \in \mathcal{O}_2.attributes (\mathcal{A}_1 \equiv \mathcal{A}_2 \wedge \iota(\mathcal{A}_1) = \iota(\mathcal{A}_2)) \wedge \\
 &\forall \mathcal{A}_2 \in \mathcal{O}_2.attributes \\
 &\exists \mathcal{A}_1 \in \mathcal{O}_1.attributes (\mathcal{A}_2 \equiv \mathcal{A}_1 \wedge \iota(\mathcal{A}_2) = \iota(\mathcal{A}_1))
 \end{aligned} \tag{5.12}$$

Equation 5.13 illustrates equivalence between two `OBJECTDEFINITIONS` $\mathcal{D}_1^O, \mathcal{D}_2^O$. Note that not only the `OID`, and names of the `OBJECTDEFINITION` and respective `ATTRIBUTEDEFINITION` are compared but also the order of the `ATTRIBUTEDEFINITIONS` within both `OBJECTDEFINITIONS`.

$$\begin{aligned}
 \mathcal{D}_1^O \equiv \mathcal{D}_2^O &\Leftrightarrow \mathcal{D}_1^O \cong \mathcal{D}_2^O \wedge \\
 &\mathcal{D}_1^O.name = \mathcal{D}_2^O.name \wedge \\
 &\forall \mathcal{D}_1^A \in \mathcal{D}_1^O.attributeDefinitions \\
 &\exists \mathcal{D}_2^A \in \mathcal{D}_2^O.attributeDefinitions (\mathcal{D}_1^A \equiv \mathcal{D}_2^A \wedge \\
 &\quad \iota(\mathcal{D}_1^A) = \iota(\mathcal{D}_2^A)) \wedge \\
 &\forall \mathcal{D}_2^A \in \mathcal{D}_2^O.attributeDefinitions \\
 &\exists \mathcal{D}_1^A \in \mathcal{D}_1^O.attributeDefinitions (\mathcal{D}_2^A \equiv \mathcal{D}_1^A \wedge \\
 &\quad \iota(\mathcal{D}_2^A) = \iota(\mathcal{D}_1^A))
 \end{aligned} \tag{5.13}$$

5.2.5 Merging Models and Metamodels

Next to detailing the design of the differencing, we explain our approach to merge different models. We already outlined that an information source has an import model which is a branch of the EA repository's model (cf. Section 5.2.1). We assume each import model conforms to an import metamodel which in turn is an instance of the metamodel introduced in Section 5.1.2. In addition, we assume that each import metamodel is a branch of an EA metamodel (cf. Definition 5.2).

Before we introduce prerequisites to build an understanding for our subsequent considerations, we vivify our approach to merge models with a brief example. Figure 5.18 depicts

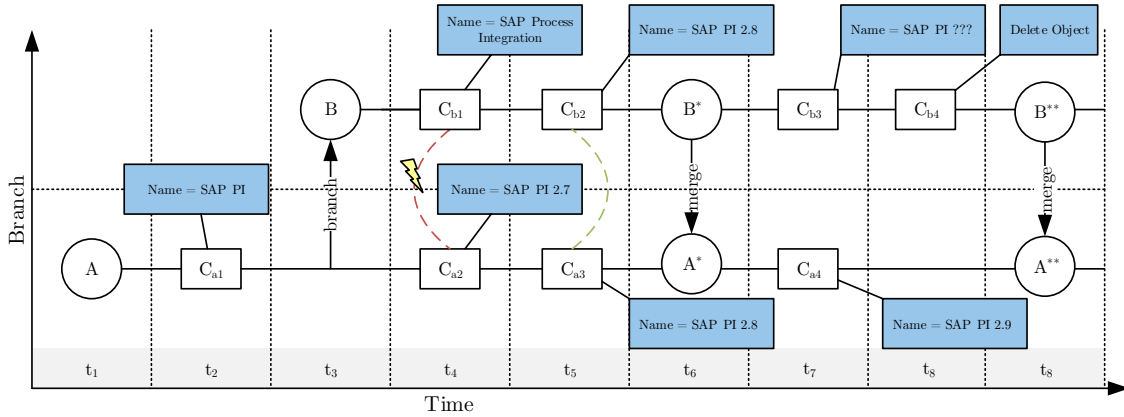


Figure 5.18: Different evolution paths with the same resulting state

two different branches of model elements denoted A and B with changes $C_{a1..n}$ and $C_{b1..n}$ respectively. Branch B is created at time t_3 . Thereafter, it evolves independently such that at t_4 both branches apply different changes to model elements that describe the same real-world circumstances. However, at t_5 both branches are equal again. Next, we merge the evolved branch B^* with A^* at t_6 . Since we chose an operation-based approach, a naïve approach would assume to compare all changes since the creation of the branch must be considered to detect conflicts, i.e. C_{b1} and C_{a2} would certainly raise a conflict while C_{b2} and C_{a3} would not. Following the Atomicity, Consistency, Isolation, Durability (ACID) paradigm, one could even propose to compare C_{b1} and C_{a3} as well as C_{b2} and C_{a2} . The resulting conflicts would certainly not reflect the intention one has in mind when merging two models with each other. On the other hand, just looking at the latest state of a model has clear drawbacks (cf. *state-based* in Section 2.3.3).

We claim that just looking at the latest revision, i.e. a state-based comparison, is insufficient, in particular considering the relationship of OBJECTS and ATTRIBUTES with their respective definitions. This hypothesis is supported by other researchers: For instance, Koegel et al. [KHL⁺10] found that the size and types of changes are relevant factors for understanding changes. The authors state that additional information recorded by an operation-based approach can be valuable.

Our approach to merge models is first *identifies* CHANGESSETS which are in an eventual conflicting state since their MODELELEMENTS are related to each other and then *compresses* changes. For our merge scenario illustrated in Figure 5.18, that means changes C_{a2} and C_{b1} get superseded by their respective successors C_{b2} and C_{a3} .

For our next considerations, let us assume an enterprise architect discusses with a data owner about the system described by this model element. During the discussion, the exact version of that system has been unclear. As a reaction and to prevent any decision based on uncertain information, the enterprise architect changed the name indicating the version. To indicate that the version is unclear, the enterprise architect just appends ‘???’ to the element in branch B^* at t_7 . During a quality initiative at t_8 , the entire OBJECT gets deleted accidentally. Meanwhile in branch A^* the correct version number is added, i.e. the respective ATTRIBUTE has been updated at t_7 . Again, a naïve approach that only

considered CHANGESETS that are applied on one kind of MODELELEMENTS, in this example ATTRIBUTES, would not detect such a conflict. The structural relationship between the OBJECT and its ATTRIBUTES must be considered. This holds true for similar relationships as we detail below.

In the following, we explain prerequisites for a merge algorithm that builds a central element of MODELGLUE. It can be divided in the following phases:

1. identification of potential collisions and selection of involved CHANGESETS, i.e. only those elements that exert influence on each other are analyzed,
2. compression of atomic operations, i.e. all CHANGES within the involved CHANGESETS are compressed,
3. conflict detection and classification, i.e. which CHANGES are actually conflicting,
4. creation of a preview model that contains the tentative merge result, i.e. apply non-conflicting changes to a temporary model that serves for the conflict resolution, and
5. conflict task generation and assignment, i.e. create tasks that instruct humans which conflicts arose, determine responsible role, and send the task to this role.

We provide details to foster an understanding of this five-phased approach and subsequently present the algorithm.

5.2.5.1 Selecting Potentially Conflicting Changesets

Kirschner [Ki14, pp.31–32] provides insights how to reconcile the *baseline time* t_b of a branch. Informally, it denotes the time a model has been branched from another model, e.g. in Figure 5.18 $t_b = t_3$.

For an identification of potential collisions of changes, we distinguish between OBJECTS, ATTRIBUTES, OBJECTDEFINITIONS, and ATTRIBUTEDEFINITIONS. As introduced by the metamodel in Section 5.1.2, we can access each of these MODELELEMENTS through the association between CHANGESET and MODELELEMENT. Let us assume we analyze two CHANGESETS which both access two MODELELEMENTS.

For our following considerations, we further assume that the set of directly compared CHANGESETS contains only CHANGESETS that refer to MODELELEMENTS with the same classifier. Put differently, we only compare changes of OBJECTS with other changes on OBJECTS, changes on ATTRIBUTES with other changes on ATTRIBUTES, changes on ATTRIBUTEDEFINITIONS with changes on ATTRIBUTEDEFINITIONS, and changes on OBJECTDEFINITIONS with other changes of OBJECTDEFINITIONS. In turn, that does not mean, we do not intend to analyze the context of a MODELELEMENT, e.g. the OBJECTDEFINITION of an OBJECT.

More formally, for two given Objects $\mathcal{O}_1, \mathcal{O}_2$, we analyze changes iff Equation 5.14 holds.

$$\begin{aligned}
 \mathcal{O}_1 \not\sim \mathcal{O}_2 &\Leftrightarrow \mathcal{O}_1 \cong \mathcal{O}_2 \vee \\
 &(\mathcal{O}_1.\text{name} = \mathcal{O}_2.\text{name} \wedge \\
 &(\mathcal{O}_1.\text{parent} \cong \mathcal{O}_1.\text{parent} \vee \mathcal{O}_1.\text{parent}.\text{name} = \mathcal{O}_1.\text{parent}.\text{name}) \vee \\
 &(\mathcal{O}_1.\text{objectDefinition} \cong \mathcal{O}_2.\text{objectDefinition} \vee \\
 &\mathcal{O}_1.\text{objectDefinition}.\text{name} = \mathcal{O}_2.\text{objectDefinition}.\text{name}))
 \end{aligned} \tag{5.14}$$

That is, we do not compare the entire object (cf. also Equation 5.12 on p.167), but we compare their OIDs and UIDs. Although that is the common case, other scenarios must be taken care of. Only if the OBJECTS are not congruent, their names are analyzed. Within a certain scope, these must be unique otherwise they describe the same OBJECT. The first case described by Equation 5.14 is the scope of the same parent OBJECT. Thus, the parent's OIDs and UIDs must be analyzed to ensure that they are not branches of each other. The same name of the parent could also pose a problem. The second case is an OBJECT of the same kind, i.e. OBJECTS conforming to the same OBJECTDEFINITION, may be involved in a conflict if they possess the same name. This serves to ensure that a different OBJECT has not been renamed and is used as the parent for this object.

And for ATTRIBUTES $\mathcal{A}_1, \mathcal{A}_2$, we analyze CHANGESETS if Equation 5.15 holds.

$$\begin{aligned}
 \mathcal{A}_1 \not\sim \mathcal{A}_2 &\Leftrightarrow \mathcal{A}_1 \cong \mathcal{A}_2 \vee \\
 &\mathcal{A}_1.\text{name} = \mathcal{A}_2.\text{name} \wedge \\
 &\mathcal{A}_1.\text{object} \cong \mathcal{A}_2.\text{object}
 \end{aligned} \tag{5.15}$$

The name of an OBJECTDEFINITION within a model must be unique, consequently congruence is not the only possibility for a collision. More formally, we write for two OBJECTDEFINITIONS $\mathcal{D}_1^{\mathcal{O}}, \mathcal{D}_2^{\mathcal{O}}$:

$$\mathcal{D}_1^{\mathcal{O}} \not\sim \mathcal{D}_2^{\mathcal{O}} \Leftrightarrow \mathcal{D}_1^{\mathcal{O}} \cong \mathcal{D}_2^{\mathcal{O}} \vee \mathcal{D}_1^{\mathcal{O}}.\text{name} = \mathcal{D}_2^{\mathcal{O}}.\text{name} \tag{5.16}$$

Similarly, the name of an ATTRIBUTEDEFINITION within an OBJECTDEFINITION must be unique, consequently for two ATTRIBUTEDEFINITIONS $\mathcal{D}_1^{\mathcal{A}}, \mathcal{D}_2^{\mathcal{A}}$, we write:

$$\begin{aligned}
 \mathcal{D}_1^{\mathcal{A}} \not\sim \mathcal{D}_2^{\mathcal{A}} &\Leftrightarrow \mathcal{D}_1^{\mathcal{A}} \cong \mathcal{D}_2^{\mathcal{A}} \vee \\
 &\mathcal{D}_1^{\mathcal{A}}.\text{name} = \mathcal{D}_2^{\mathcal{A}}.\text{name} \wedge \\
 &\mathcal{D}_1^{\mathcal{A}}.\text{objectDefinition} \cong \mathcal{D}_2^{\mathcal{A}}.\text{objectDefinition}
 \end{aligned} \tag{5.17}$$

So far, we assumed that only changes of the same kind of MODELELEMENT are considered. Now we define some exceptions. The first exception is that we must investigate concurrently performed changes on OBJECTS and their OBJECTDEFINITIONS, i.e. they are related and may exert influence on each other. For our next considerations, let us assume an OBJECT \mathcal{O} with an OBJECTDEFINITION $\mathcal{D}^{\mathcal{O}}$.

$$\mathcal{O} \not\sim \mathcal{D}^{\mathcal{O}} \Leftrightarrow \mathcal{O}.\text{objectDefinition} \cong \mathcal{D}^{\mathcal{O}} \tag{5.18}$$

If Equation 5.18 holds, these CHANGESETS are analyzed. A potential conflict that could arise during the further analysis of these CHANGESETS is called model/metamodel conflict (see Example 4.2 on p. 67).

Our next considerations focus on CHANGES applied to an ATTRIBUTEDEFINITION \mathcal{D}^A that is part of an OBJECTDEFINITION \mathcal{D}^O which is affected by an issued CHANGE.

$$\mathcal{D}^A \hat{\bowtie} \mathcal{D}^O \Leftrightarrow \mathcal{D}^A.\text{objectDefinition} \cong \mathcal{D}^O \quad (5.19)$$

Our final concern about depending CHANGES on MODELELEMENTS can be explained with an ATTRIBUTE \mathcal{A} and ATTRIBUTEDEFINITION \mathcal{D}^A :

$$\begin{aligned} \mathcal{A} \hat{\bowtie} \mathcal{D}^A \Leftrightarrow & \mathcal{A}.\text{attributeDefinition} \cong \mathcal{D}^A \vee \\ & (\mathcal{A}.\text{name} = \mathcal{D}^A.\text{name} \wedge \\ & \mathcal{A}.\text{object}.\text{objectDefinition} \cong \mathcal{D}^A.\text{objectDefinition}) \end{aligned} \quad (5.20)$$

As illustrated in Equation 5.20, modifications on the name of an ATTRIBUTE and concurrent modifications on an ATTRIBUTEDEFINITION are investigated, too. Note that VALUES are not considered to be MODELELEMENTS (cf. Figure 5.2 on p. 134) and are stored in this step as CHANGESETS that address ATTRIBUTES. During the conflict detection however, attributes and values are considered separately⁴. For the further considerations, detailed in the next sections, we introduce Definition 5.5 to refer to CHANGESETS of related MODELELEMENTS.

DEFINITION 5.5: Potentially conflicting changesets

We call CHANGESETS $\mathcal{C}_1, \mathcal{C}_2$ potentially conflicting iff Equation 5.21 holds.

$$\mathcal{C}_1 \hat{\bowtie} \mathcal{C}_2 \Leftrightarrow \mathcal{C}_1.\text{modelElement} \hat{\bowtie} \mathcal{C}_2.\text{modelElement} \quad (5.21)$$

Note that for each subclass of MODELELEMENT the respective rules (Equation 5.14–5.20) apply for the identification of related elements. ■

As of now, we provided a formal description to identify CHANGESETS referring to related elements. This reduces the number of CHANGESETS considerably and leaves us with CHANGESETS which potentially are conflicting. In the following, we have a closer look at CHANGESETS.

5.2.5.2 Operation-based Changesets

As outlined in Chapter 3, there are fundamental differences of merge approaches following either a state-based or operation-based paradigm. In their empirical study, Koegel et al. conclude that “operation-based change tracking exhibits advantages in understanding more complex changes” [KHL⁺10]. In particular since “additional information recorded by the operation-based approach can be valuable” [KHL⁺10] and is more efficient in terms of

⁴We refer the interested reader to Kirschner [Ki14, p. 35ff] for further implementation details on the detection of operations and conflicts.

computational complexity [Me02], we choose an operation-based tracking of CHANGES to MODELELEMENTS. Moreover, it is important to be able to determine *who* changed which particular MODELELEMENTS at a certain point in time. Such information could be highly relevant not only in the course of an audit trail but also to find the relevant persons to talk to and to collaborate with when conflicts in an EA model occur. Subsequently, we outline our notion of operation-based CHANGESETS briefly.

In MODELGLUE, operations applied on MODELELEMENTS are stored in form of CHANGESETS. Thereby, an element always carries the most recent CHANGES of CHANGESETS in a persistent manner (cf. *delta-backward* in Section 2.3.9) whereas all CHANGESETS related to a MODELELEMENT can be seen as a version history. In our metamodel, a CHANGESET (cf. Section 5.1.2) summarize user transactions whereas CHANGES are atomic modifications to mutable objects, i.e. MODELELEMENTS.

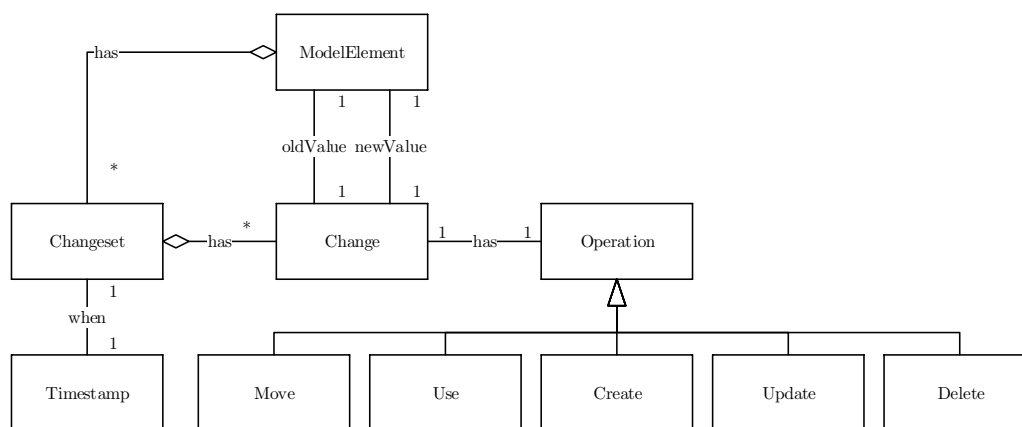


Figure 5.19: Operation-based tracking of modifications using CHANGESETS and CHANGES

Figure 5.19 depicts details of the concepts CHANGESET and CHANGE. The entire transaction is recorded as a CHANGESET issued by a particular user at a particular time (cf. Figure 5.2 on p. 134). Each CHANGE is made to exactly one MODELELEMENT and has an old as well as a new value. The OPERATIONS that can be applied to a MODELELEMENT are as follows:

Create induces a new MODELELEMENT from scratch. Newly created MODELELEMENTS have separate UIDs and an OID that refers to this UID. This initial condition between OID and UID denotes that each MODELELEMENT is initially its own origin.

Update alters an existing MODELELEMENT. For instance, a change of an OBJECT's name, or a change of its ATTRIBUTES.

Delete removes an existing MODELELEMENT. This operation does not remove any data physically. On the one hand, this guarantees traceability of CHANGES for an eventual audit trail. On the other hand, it may facilitate conflict detection as deleted MODELELEMENTS could be common origin of other MODELELEMENTS. If elements are discarded physically, one has to keep track of the deletions to identify potential conflicts in the course of a merge.












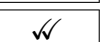
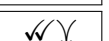
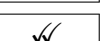
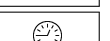
Move denotes a change in the hierarchy of the model. Since MODELGLUE does not support inheritance, move operations have considerably less impact than in a systems supporting inheritance. In such a system, the move of a supertype is regarded a non-trivial conflict situation with considerably high semantics (cf. [WLS⁺12]).

Use is a passive operation to a MODELELEMENT \mathcal{E}_a , i.e. it is not modified by the use operation. This subtype of a CHANGE is created as a side effect of a CREATE or an UPDATE to another MODELELEMENT \mathcal{E}_b referencing \mathcal{E}_a , i.e. updating or creating a link.

5.2.5.3 Compressing Changes

As discussed above, CHANGESETS can be regarded transactions incorporating multiple CHANGES. Some of these CHANGES are reciprocally rectified; for instance, multiple UPDATES on the same ATTRIBUTE get compressed and the most recent UPDATE prevails. Other operations cancel each other out; for instance, multiple UPDATES and a subsequent DELETE. The DELETE prevails if it is the most recent CHANGE. Table 5.2 gives a more structured view how we compress CHANGES. Note that only CHANGES within a branch can be compressed, i.e. inter-branch compression requires a conflict detection first. In fact, we only compress CHANGES prior to the conflict detection. This way, compressing CHANGES preserves all information to resolve conflicts manually. Similarly, we do not provide an order in which CHANGES would become executable since this could obscure the original intentions. Instead, we present conflicting CHANGES in their chronological order (cf. Section 5.3.2.4).

In our implementation, a subsequent UPDATE operation with a preceding CREATE gets reduced to a CREATE operation; however, this is system specific since a CREATE in MODELGLUE first creates an empty MODELELEMENT and subsequently performs an UPDATE on this element to set its initial name.

	Create	Delete	Update	Use	Move
Create					
Delete					
Update					
Use					
Move					




 take most recent change
 take both
 only if from two different source objects

Table 5.2: Outline of the compression method applied to changes

5.2.5.4 An Algorithm for Model Merging

Assuming that models \mathcal{M}^a , \mathcal{M}^b build a federation, i.e. $\mathcal{M}^a \xrightarrow{\circ} \mathcal{M}^b$ (cf. Definition 4.5) and at the same time the models are branches of each other, i.e. $\mathcal{M}^{a \rightarrow b}$ (cf. Definition 5.2), Algorithm 4 details how different models in Federated EA Model Management can be merged such that responsible ROLES, i.e. humans, are notified via TASKS that carry contextual information of a conflict. This information is meant to facilitate the conflict resolution. Thereby, \mathcal{M}^a is denoted the model of an information source whereas \mathcal{M}^b constitutes the EA model. In the following, we describe the algorithm in more detail.

Lines 1–2: In a first step, for each task type, a Conflict Information Container (CIC) is created that serves as a temporary in-memory data structure to store the relevant meta-information for a task. The CIC data structure is described by Kirschner who introduces the concept of the CIC to cope with multiple conflicts per MODELELEMENT. We refer the interested reader to [Ki14, pp. 24,44,48] for a detailed description how this structure can be employed to not only describe conflicts, but also to detect and store n-way conflicts based on the algorithm described in this thesis. Additionally, line 2 serves to allocate a collection that temporarily saves the CHANGESETS under investigation.

Line 3–7: Next, we analyze all CHANGESETS that were created since the baseline time t_b . Note that we only add those CHANGESETS for further analysis that describe changes on MODELELEMENTS, which are related to each other (cf. Section 5.2.5.1). Thereafter, the resulting CHANGESETS contain only potentially conflicting changes.

Line 8–9: Subsequently, the algorithm takes all CHANGES within the identified CHANGESETS and compresses them, i.e. some CHANGES within one branch eliminate each other such that they can be compressed. Details of this compression are described in Section 5.2.5.3.

Line 10–25: Prior to the actual conflict detection, we ensure that only potentially conflicting CHANGESETS are checked in subsequent steps (lines 11–12). For the sake of brevity and to increase readability, we used only two `foreach` loops to sketch the core idea of the algorithm whereas the actual implementation requires four loops. The actual conflict detection is performed through a lookup in the matrix of the conflict resolution strategy (line 14). Thereby, the OPERATION performed and the classifier of a MODELELEMENT serve to lookup if and what further actions are necessary. Classifier denotes the type of the subclass. The matrix returns either one of the tasks approve, resolve, or validate or a customized merge rule that is part of an organization-specific merge strategy. Each cell within the matrix specifies 1 of 210 cases for the conflict detection (cf. Section 5.2.5.6). In line with the selection of the CHANGESETS, the detection embraces elements of the model as well as the metamodel and their relationships to each other (cf. Definition 5.5).

Line 14–25: The algorithm looks up the conflict resolution strategy that is configured in MODELGLUE to resolve a certain conflict. The information, i.e. CHANGESETS and type of conflict is added to the respective CIC data structure. If no further action is configured (line 25), the CHANGES are not in conflict and no further action is required.

Algorithm 4: n-way merging of models based on the initial versions published in [KR14, Ki14]

Input: Source Models $\mathcal{M}_{1..n}$, Baseline Time t_b , Conflict Resolution Strategy ς , Tasks $T_{1..n}^i$
Output: Task $T_{1..n}$, Target Model \mathcal{M}_t

```

1: approve, resolve, validate, custom  $\leftarrow$  {key:  $\emptyset$ , value:  $\{\emptyset\}$ }
2: changesets  $\leftarrow$   $\{\emptyset\}$ 
3: // Phase 1: select and compress changes
4: foreach  $\mathbb{C}_1 \in \mathcal{M}_{1..n}.modelElements.changesets(\mathbb{C}_1.when > t_b)$  do
5:   foreach  $\mathbb{C}_2 \in \mathcal{M}_{1..n}.modelElements.changesets(\mathbb{C}_2.when > t_b)$  do
6:     if  $\mathbb{C}_1 \hat{\vee} \mathbb{C}_2$  then
7:       changesets  $\leftarrow$  changesets  $\cup$   $\mathbb{C}_1 \cup \mathbb{C}_2$ 

8: // Phase 2: compress changes
9: changesets  $\leftarrow$  compressOperations(changesets)
10: // Phase 3: detect conflicts
11: foreach  $\mathbb{C}_1 \in changesets, \mathbb{C}_2 \in changesets$  do
12:   foreach  $\delta_1 \in \mathbb{C}_1.changes, \delta_2 \in \mathbb{C}_2.changes(\mathbb{C}_1 \hat{\vee} \mathbb{C}_2)$  do
13:     // perform lookup of rule in conflict resolution strategy
14:     conflict  $\leftarrow$ 
15:      $\varsigma[\mathbb{C}_1.modelElement.classifier][\delta_1.operation][\mathbb{C}_2.modelElement.classifier][\delta_2.operation]$ 
16:     switch conflict.classifier do
17:       case approve
18:         approve  $\leftarrow$  appendToCIC(approve,  $\delta_1, \delta_2, \mathbb{C}_1, \mathbb{C}_2$ )
19:       case resolve
20:         conflicts  $\leftarrow$  appendToCIC(conflicts,  $\delta_1, \delta_2, \mathbb{C}_1, \mathbb{C}_2$ )
21:       case validate
22:         validate  $\leftarrow$  appendToCIC(validate,  $\delta_1, \delta_2, \mathbb{C}_1, \mathbb{C}_2$ )
23:       case custom
24:         custom  $\leftarrow$  appendToCIC(custom,  $\delta_1, \delta_2, \mathbb{C}_1, \mathbb{C}_2$ )
25:       else
26:         continue

26: // Phase 4: create 'tentative' merge result
27: foreach  $\mathcal{E} \in (resolve.keys \cup approve.keys \cup validate.keys \cup custom.keys)$  do
28:   foreach  $\mathbb{C} \in changesets$  do
29:     if  $\mathcal{E}.uid \equiv \mathbb{C}.modelElement.uid$  then
30:       changesets  $\leftarrow$  changesets  $\setminus$   $\mathbb{C}$ 

31:  $\mathcal{M}_t \leftarrow$  getBaselineModel( $\mathcal{M}_{1..n}, t_b$ )
32: foreach  $\mathbb{C} \in changesets$  do
33:    $\mathcal{M}_t \leftarrow \mathcal{M}_t \cup \mathbb{C}$ 

34: // Phase 5: create conflict resolution tasks and invoke custom merge rules
35: foreach  $cic \in approve.values$  do
36:   owners  $\leftarrow$   $\{\emptyset\}$ 
37:   foreach entry  $\in cic.entries$  do
38:     owners  $\leftarrow$  owners  $\cup$  entry.changeset.modelElement.responsibleRole
39:     // see Section 5.1.2.4 for further details on how to determine the responsible role for
40:     // a model element
41:   if  $cic \notin (T_{1..n}^i).cics$  then
42:      $T_i \leftarrow$  new Approve( $cic, owners$ )

42: // repeat from line 35 for 'resolve' and 'validate' tasks
43: foreach  $cic \in custom.values$  do
44:   if  $cic \notin (T_{1..n}^i).cics$  then
45:      $cic.invokeRule()$ 

```

Line 26–33: CHANGESETS which are contained in a CIC are removed from the set of CHANGESETS under investigation (line 33). The remaining CHANGESETS are applied to a model at time t_b which is then target model \mathcal{M}_t .

Line 35–41: In a final step, the algorithm intends to create new conflict resolution tasks, which are equipped with the meta-information contained in the CIC and sent to the responsible role (cf. also Algorithm 1 on p. 137). Thereby, we reconcile tasks that have been marked as ‘ignored’ in the course of a merge that previously took place. Only if the CIC is not found in one of these tasks $T_{1..n}^i$, new tasks are instantiated and sent to their owner (line 40). The same procedure is performed for ‘resolve’ and ‘validate’ tasks as well. The final action of the algorithm is to invoke the custom merge rules specified in the conflict resolution strategy (line 43). We assume that by executing the rule, any conflicts described by the respective CIC are resolved.

Note that in the course of a merge, the source models do not cease to exist. Hence, we merge multiple source models ‘with’ a target model, not ‘into’. This property is especially important to reconcile the identity of elements that are imported from an information source.

More sophisticated change detection, e.g. in elements that feature transitive relationships, are omitted here. We put primary focus on the conformance of OBJECTS to OBJECTDEFINITIONS and ATTRIBUTES to their ATTRIBUTEDEFINITIONS. However, our final considerations concerning the algorithm are additional checks for any relationships between the two elements that semantically affect each other by CHANGESETS \mathbb{C}_1 and \mathbb{C}_2 . We considered to define such relationships explicitly such that each relationship specified $(\mathcal{E}_1, \mathcal{E}_1)$ influences the outcome of the check $\mathbb{C}_1 \hat{\gamma} \mathbb{C}_2$.

5.2.5.5 Task Types in Federated EA Model Management

In [RHM13a], we outlined an initial version of different task types. In the following, we describe and refine the different types of these tasks. We distinguish between the following types of tasks:

Assign Role is a TASK type concerned with the assignment of the responsible role, readers, and writers of a particular OBJECTDEFINITION, OBJECT, ATTRIBUTEDEFINITION, or ATTRIBUTE. If a suitable role is defined, tasks are sent to a role. This addressee, i.e. the responsible role, is inferred as described in Section 5.1.2.4.

Validate commonly refers to the validation of particular ATTRIBUTES or entire OBJECTS. When assigning validation tasks, CHANGESETS are already applied to the respective MODELELEMENT such that respective CHANGES must be reverted or new CHANGESETS must be issued. This can be done by any writer who are informed by default. Due to their write access, writers are immediately able to correct flaws issued by made CHANGES. If no objections are raised concerning the applied CHANGES or further actions are required, the addressee of a TASK manually marks it as ‘solved’. Commonly, validate TASK are employed to inform about concurrent CHANGES that might have a semantic impact on another CHANGE. In such a case, the receiving user has to review

the CHANGES made and validate whether applied changes are still valid in the new context or in turn are contradictory given the changed context. By default, the user does not have to modify any value.

Approve is required to approve certain modifications and always requires user intervention. By default, only one of the pending CHANGES can be approved and remaining CHANGES are discarded (cf. REVOKECHANGES in Section 6.3.2). In the course of concurrent model CHANGES, respective editors are informed. If the editors cannot determine whether to approve a CHANGE or not, the system involves the responsible ROLE, i.e. the TASK is forwarded. For instance, deletions of entire OBJECTS must be approved or CHANGES of certain ATTRIBUTES that a responsible role is accountable for, e.g. CHANGES of a ‘service level’ ATTRIBUTE. If both parties agreed upon a CHANGE, the system marks this TASK as ‘resolved’ and the respective MODELELEMENT switches its state to ‘normal’ if no further TASKS exist (cf. Section 5.1.2.5).

Resolve seeks to merge multiple CHANGES into one consistent model state. This requires user intervention to resolve a situation of concurrent CHANGES on MODELELEMENTS describing the same circumstances of the real-world or CHANGES on adjacent MODELELEMENTS that may have an impact on the semantics of another CHANGES (cf. Section 5.2.5.1).

The notion of order plays an important role when resolving a conflict. Although some approaches seek to reorder CHANGES such that they become applicable, we claim that conflict resolution requires human intervention. The software support for a conflict is limited to present the CHANGES, respective roles, and timestamp to the user in a reverse-chronological order such that the most recent CHANGE is on top of all the CHANGES.

It is perhaps the most complex TASK since multiple parties must be involved in order to decide on pending model CHANGES. To support this TASK type, we adopt the general idea of a tentative merge result (cf. Section 2.3.5) in order to store any concurrent model CHANGES. Thereafter, the system is able to show made CHANGES to the end-users with the original version, i.e. a three-way difference (cf. Section 2.3.1). In case, different CHANGES are in conflict, i.e. cannot be resolved by the conflict resolution strategy currently configured in an automated manner, users may chose which of the CHANGES they want to apply. In contrast to an approve TASK, multiple CHANGESETS may get applied or new CHANGESETS are issued by the resolving parties. This resolution process commonly takes place in a synchronous manner such that a technical solution should support face-to-face to face communication (cf. REQUIREMENT Us8 in Section 4.3.4).

Propagate refers to the (manual) propagation of CHANGES to other information systems, i.e. an integrated information source. It is generated if a CHANGESET is applied to a MODELELEMENT whose OID refers to information stored externally. This task type asks to apply CHANGES in an information source, i.e. it propagates CHANGES. Generally speaking, this propagation can be done either automatically via technical interfaces, or manually by the assigned ROLE. We propose to propagate CHANGES via human tasks. This way, an information source is integrated logically in a bidirectional manner but one does not have to cope with the complexity of developing and

maintaining a plethora of bidirectional interfaces. The URI of the OID encodes the necessary information to indicate ‘where’ the CHANGE must be applied. Ideally, a web-based system is used such that the URI can serve as an Uniform Resource Locator (URL) for a hyperlink target. Given today’s browsers, this would foster a seamless integration within a federation. Additionally, the user gets context information that describes the details of the CHANGE to be performed in an information source.

Inform tasks are no conflict tasks. They serve to inform users about automated merge actions. As soon as they are viewed, they are marked as ‘solved’ by the system automatically. Further, they are disregarded automatically after passing their due-date.

Document asks the writers to maintain a certain OBJECT or ATTRIBUTES thereof. This task is automatically sent to writers as soon as an ATTRIBUTE is set as strict (cf. [RHM13a]).

5.2.5.6 Conflict Detection

We already specified the notion of connectedness of MODELELEMENTS in Section 5.2.5.1. In contrast to Wieland et al. [WLS⁺12], this approach considers not only CHANGES on elements of the model but also CHANGES on elements of the metamodel. Moreover, we consider CHANGES on the metamodel that may exert (semantic) influence on the model and vice versa. This way our conflict detection is far more complex than a 5 by 5 operation-matrix (cf. Table 4.12 on p. 111).

The actual detection is performed with a LUT whereas its cells store formal specifications for the conflict detection and resolution. In his master’s thesis, Kirschner [Ki14, p. 42] reports on 325 possible cases to consider. Although his considerations include also operations on VALUES of ATTRIBUTES, the order of magnitude remains the same removing the operations for values from the equation. On the other hand, Kirschner presents both conflict resolution strategies in his thesis ([Ki14, Appendix A]). The cells, i.e. specified actions, for the tolerant conflict resolution strategy embrace only 30 conflict/approve tasks, i.e. only 30 cases in which MODELGLUE cannot decide how to merge changes. These 30 cases build the minimal set that has to be specified either by the task of the tolerant strategy or by a customized merge rule.

$$\sum_{i=1}^n i = \begin{cases} 325 & \text{if } o = 5, e = 5 \\ 210 & \text{if } o = 5, e = 4 \end{cases} \quad (5.22)$$

Equation 5.22 provides the exact numbers of cases to be considered whereas the product $n = e * o$ is defined by the cardinal numbers of involved OPERATIONS = {CREATE, UPDATE, DELETE, MOVE, USE}, $o := |\text{OPERATIONS}| \Rightarrow 5$, and considered MODELELEMENTS = {ATTRIBUTES, ATTRIBUTEDEFINITIONS, OBJECT, OBJECTDEFINITIONS}, $e := |\text{MODELELEMENTS}| \Rightarrow 4$.

Since it is beyond the scope of the present thesis to specify 325 cases formally, we outline the general principles, we followed implementing both strategies, i.e. tolerant and strict (cf. Section 5.2.8). Further, we provide insights on how to specify organization-specific

merge rules that are executed instead of a task. In the remainder of this section, we provide designs for intuitive UIs that allow users to specify merge rules.

5.2.6 Manual Detection of Conflicts

Up to now, we discussed automated conflict detection as well as means for the assessment of quality via differencing, and conflict resolution. In this section, we discuss the role of model conflicts or other issues in models that cannot be detected automatically. In [RHM13b], we report on phenomena between different models and coined the term *abstraction gap*. In the following, we explain the notion of an abstraction gap more formally and present means to overcome it. In this vein, we present UI support for the manual creation of tasks.

5.2.6.1 Abstraction Gaps Between Models

In [RHM13b], we use the ArchiMate 2.0 metamodel as an illustrating example to show how to overcome an abstraction gap employing an interactive visualization that propagates visual changes to the EA model. Before we continue to explain the details of an abstraction gap, let us start with an example.



EXAMPLE 5.5: An abstraction gap in an EA model

A CMDB contains information about NODES. According to the ArchiMate 2.0 specification, a NODE DEVICE is “a hardware resource upon which artifacts may be stored or deployed for execution. A device is a specialization of a node that represents a physical resource with processing capability. It is typically used to model hardware systems such as mainframes, PCs, or routers” [Th12b, p. 46].

An additional information source, i.e. an ESB namely SAP Process Integration (PI), contains information about the APPLICATION COMPONENT and their communication. Many EA stakeholders raise concerns when it comes to finding out which NODE DEVICE is used to run an APPLICATION COMPONENT. Especially knowledge of the transitive relationships of APPLICATION COMPONENTS could be verified with runtime information of the NODE DEVICES. Currently, the EA metamodel does not include NODE DEVICES. To address EA stakeholder demands, the EA coordinator wants to integrate information about NODE DEVICES in the EA model. However, by including the NODE and its subclasses, the EA modeling expert diagnoses an arising problem. Currently, NODE instances are not mapped to APPLICATION COMPONENT instances. This piece of information is missing as well as it is not captured in the respective metamodels, i.e. there is an abstraction gap between NODE and APPLICATION COMPONENT. Although these concepts are semantically related, currently no modeling community includes this part of reality in their models.

Figure 5.20 illustrates the current situation and sketches a solution on the metamodel level. A NODE is exposed to the application layer via an INFRASTRUCTURE INTERFACE (see also [Th12a, pp. 48–49]). These INFRASTRUCTURE

INTERFACE instances are then assigned to APPLICATION COMPONENTS by which they are used, i.e. we can use the INFRASTRUCTURE INTERFACE as a binding element between NODES and APPLICATION COMPONENTS [RHM13b].

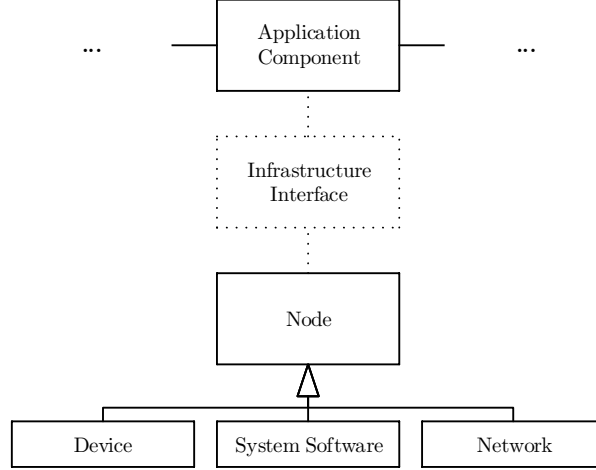


Figure 5.20: Abstraction gap within an EA model

Building on our current understanding of a federation, we describe an abstraction gap more formally. In Section 4.1.3 we established that different modelers may describe different universes. Further, we gave a formal description of the relationship of models within a federation.

DEFINITION 5.6: Abstraction gap

Let there be federation $\mathcal{M}_{b,c} \xrightarrow{\circ} \mathcal{M}_a$ with universes $\mathbb{U}_{a,b,c}$ as the perceived part of the real-world by respective roles $\rho_{a,b,c}$ whose intention is to use models $\mathcal{M}_{a,b,c}$ to describe their universe. Then, an abstraction gap in model \mathcal{M}_a occurs if and only if the federation exhibits two isolated model elements $\mathcal{E}_a \in \mathcal{M}_b$ and $\mathcal{E}_b \in \mathcal{M}_c$ (cf. Definition 4.6) for which the following holds:

- $\rho_{a,b,c}$ agree that \mathcal{E}_a and \mathcal{E}_b are semantically related,
- \mathcal{E}_a and \mathcal{E}_b are syntactically or structurally not related to each other,
- their proximity can be determined with a utility function \mathcal{R} deducing proximity relationships among model elements,
- \mathcal{R} can be described under the assumption that $\rho_{a,b,c}$ collaborate,
- \mathcal{R} requires the introduction of an additional model element \mathcal{E}_δ (cf. Equation 5.23) to describe their relationship formally.

$$\exists \mathcal{E}_\delta \in \mathcal{M}_a^* \mathcal{R}(\mathcal{E}_a, \mathcal{E}_\delta) \wedge \mathcal{R}(\mathcal{E}_\delta, \mathcal{E}_b) \quad \mathcal{E}_\delta \notin \mathcal{M}_{a,b,c} \quad (5.23)$$

■

Note that \mathcal{E}_δ in Definition 5.6 can be

- an attribute directly linking \mathcal{E}_a and \mathcal{E}_b realizing a one-to-many or many-to-one relationship,
- an object realizing a many-to-many relationship between \mathcal{E}_a and \mathcal{E}_b , or
- a model realizing transitive relationships between \mathcal{E}_a and \mathcal{E}_b .

Further, the abstraction gap also needs to be conceptualized in the respective metamodel of the federation such that new `ATTRIBUTEDEFINITIONS` and `OBJECTDEFINITIONS` must be introduced.

In line with Stachowiak [St73, pp. 131ff], we assume through the art of modeling an ‘improved’ model \mathcal{M}_a^* can be created in the sense that it provides some deeper understanding of the original, i.e. $\mathbb{U}_{a,b,c}$. As a consequence, we assume there exists at least one role that is able to close the abstraction gap providing new insights to other roles.

Sheth and Kashyap [SK93] observed similar phenomena in the domain of multidatabase systems and called it ‘*Abstraction Level Incompatibility*’. They use ‘*ANY*’ to denote that any abstraction, e.g. aggregation, generalization, specialization, can be used to define a mapping between two semantically related objects. Further, they describe the characteristics of ‘*NONE*’; it denotes that there does *not* exist a semantic relationship between objects. Finally, they also describe ‘*NEG*’ to denote that a mapping is not possible between two semantically *unrelated* objects. They further coin the term of ‘*semantic proximity*’ to describe the semantic relationship between two objects. While we observed an abstraction gap between different EA layers (cf. Section 2.1) many times in organizations, we do not claim to have found an exhaustive list of phenomena between models and, thus, refer the interested reader to Sheth and Kashyap [SK93] for further discussions of similarities between two objects.

5.2.6.2 Resolving Abstraction Gaps

To overcome an abstraction gap as described in Definition 5.6 we extend \mathcal{M}_a , such that an improved model emerges with $\mathcal{M}_a^* \subseteq \mathcal{M}_a \cup \mathcal{M}_b \cup \mathcal{M}_c \cup \mathcal{E}_\delta$. We claim that an abstraction gap *cannot* be detected automatically. However, in [RHM13b], we propose a means to overcome an abstraction gap. It involves the manual creation and configuration of tasks.

Figure 5.21 shows a preliminary version of `MODELGLUE` which is presented in [RHM13b] and [HRP⁺13b]. Although this version served for a preliminary evaluation of the prototype, the basic support for tasks has not been changed. It depicts the worklist for each individual user (❶ in Figure 5.21) as well as all tasks that are attached to a *workspace* (❷ in Figure 5.21). A workspace in this context can be considered as a model. The UI presents open and finished tasks (❸ in Figure 5.21) as a list. It has been adapted with respect to the conflict tasks described in Section 5.2.5.5. Each task can be executed, marked as finished, commented, forwarded, ignored, or deleted (❹ in Figure 5.21). `MODELGLUE` allows manual intervention not only during conflict detection but also during normal operations. That embraces the manual creation of tasks which may be required in the course of conflicts in a model that

5. Federated EA Model Management Design

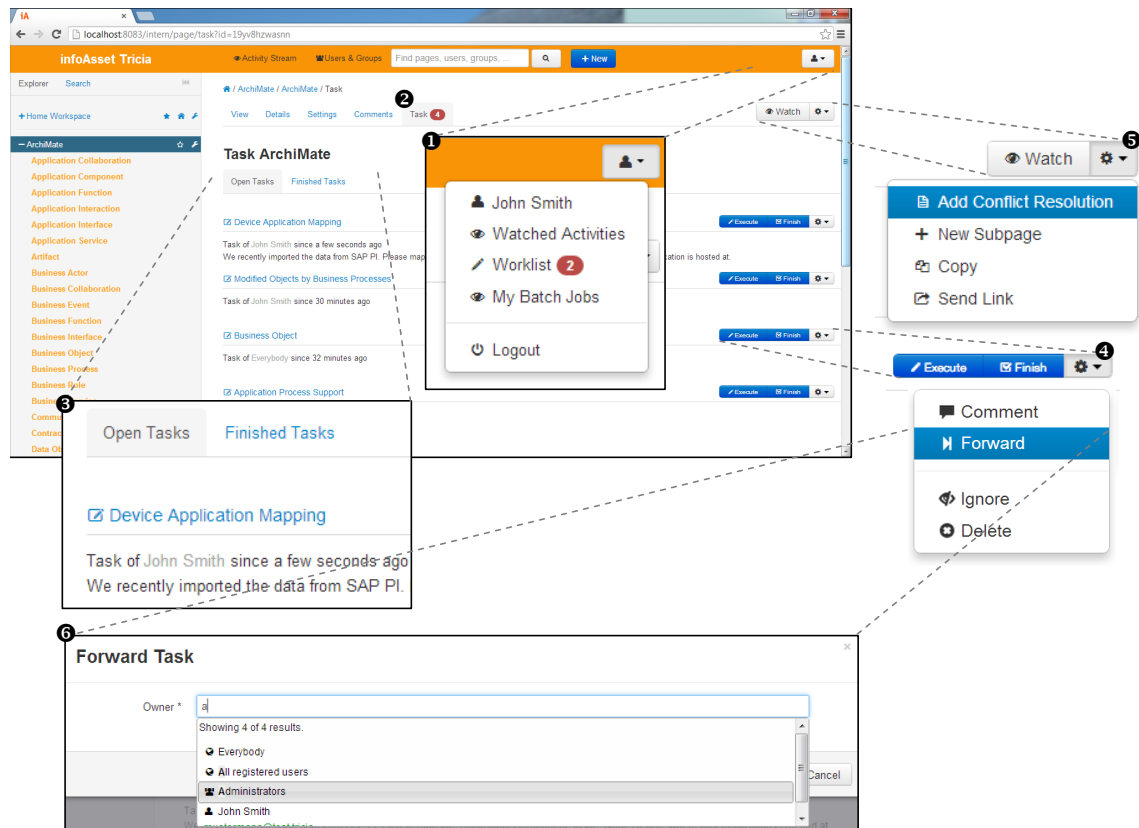


Figure 5.21: User interface of MODELGLUE: interactions with the worklist and tasks

cannot be detected automatically (⑤ in Figure 5.21). The task can be forwarded to roles which is either a single person or a group (⑥ in Figure 5.21).

Next to this general understanding of tasks (cf. also Section 5.1.2.5 and Section 5.2.5.5), we introduce one instance for a task that is created manually by an EA repository manager or an EA modeling expert. This task type is meant to overcome an abstraction gap hence, in [RHM13b] we introduce the concept of an *Abstraction Gap Resolver*. Figure 5.22 shows how to configure such an Abstraction Gap Resolver. The configuration contains information that helps to resolve the abstraction gap described in Example 5.5 on p. 179. To recap, the respective model conflict in this example is the missing mapping between DEVICES and APPLICATION COMPONENTS since this information cannot be retrieved from neither, the CMDB or the ESB, automatically.

Besides a name (① in Figure 5.22), each task has an owner (② in Figure 5.22). Specific for this task type is the embedded configuration of an Abstraction Gap Resolver. It requires to specify the two MODELELEMENTS involved in the abstraction gap, i.e. ‘Application Component’ (③ in Figure 5.22) and ‘Device instances’ (④ in Figure 5.22), as well as \mathcal{E}_δ (⑤ in Figure 5.22) which is ‘Infrastructure Element’. Since the task addresses humans, it contains a brief description that gives information about what the issue is and what has to be done in order to resolve it (⑥ in Figure 5.22).

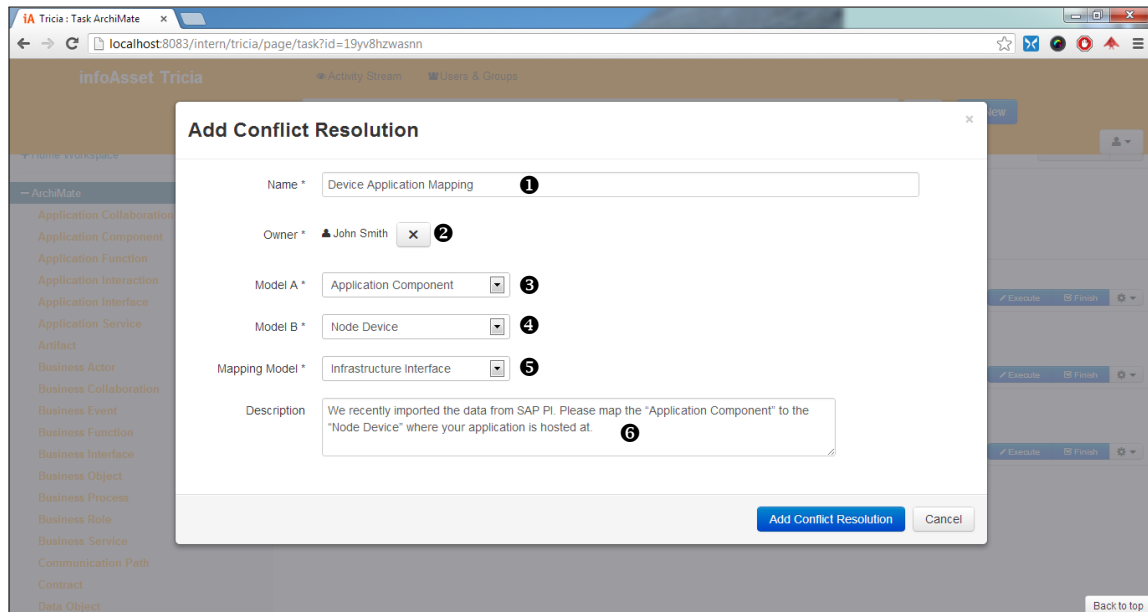


Figure 5.22: User interface of MODELGLUE: configuring an abstraction gap resolution task

In [RHM13b] we coin the term Abstraction Gap Resolver for this type of visualization. It is an interactive visualization that can be manipulated by the end-user whereas visual manipulations are propagated to the EA repository. It is a means to overcome an abstraction gap and can be used to create relationships among model elements via drag & drop operations performed within the visualization.

In the following, we introduce a conflict resolution visualization to resolve an abstraction gap based on an example given by model transformations we performed in our joint research with Buschle et al. [BEG⁺12] constituting a resolution of the abstraction gap outlined in Example 5.5 on p. 179.

Figure 5.23 depicts the resulting visualization that is shown to the addressee of a task, i.e. its owner, as presented in [RHM13b]. APPLICATION COMPONENT instances (see ❶ in Figure 5.23) and Device instances (see ❷ in Figure 5.23) are shown to the end-user. The entity INFRASTRUCTURE ELEMENT instance (see ❸ in Figure 5.23) holds the mapping information and is configured by the EA repository manager when this manual conflict resolution task is created, in order to overcome an abstraction gap between these two MODELELEMENTS APPLICATION COMPONENT and NODE DEVICES.

We assume that after the EA repository manager created the configuration for the visualization, it is forwarded to the responsible data owner or an EA stakeholder if the data owner is unable to resolve the conflict (cf. Section 5.2.7). Any existing mappings within the EA repository are illustrated as tuples (see ❹ in Figure 5.23) of n Application Component objects (see ❺ in Figure 5.23) and m Device objects (see ❻ in Figure 5.23).

The illustrated visualization type builds upon a so-called cluster map (cf. e.g. [Ma08, p. 530]) that looks familiar to EA stakeholders. In contrast to mere analysis support, this visualiza-

5. Federated EA Model Management Design

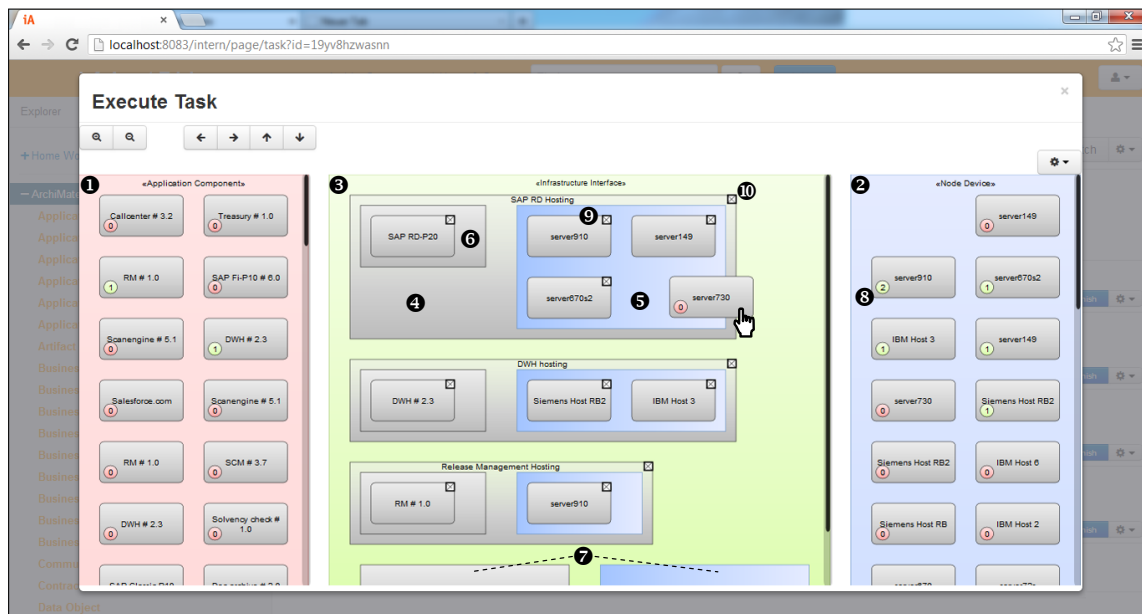


Figure 5.23: User interface of MODELGLUE: resolving an abstraction gap (annotations added to the screenshot as presented in [RHM13b])

tion type can be utilized to manipulate illustrated relationships. This way, we enable EA stakeholders and data owners to understand and contribute to the conflict resolution.

The involved roles can drag & drop elements in the interactive visualization from their type containers (see 1 and 2 in Figure 5.23) onto the respective left and right sides of the middle column (see 5 and 6 in Figure 5.23). Thereby, possible drop targets are highlighted according to the color of the container (cf. 2, 5, and 7 in Figure 5.23). Thereby, the particular drop target are pre-calculated such that while dragging elements no server round-trip is required realizing a high responsiveness for user interactions (cf. Section 6.3). Dropping elements, visual changes are propagated directly to the underlying model such that the user gets immediate visual feedback if an operation has been applied to the model successfully.

Each row (4 in Figure 5.23) represents a relationship where end-users can add model elements to an existing relationship, e.g. indicating that a new DEVICE is added to support a particular APPLICATION COMPONENT. Dropping an element either on the left or on the right side of the bottom element in the middle column (7 in Figure 5.23), a new INFRASTRUCTURE INTERFACE instance is created with the respective relationship.

While in most cases the name of an element is sufficient context information, an end-user may require additional information on an element. By clicking on the element's label, a separate window to provide this context information is opened. A counter (9 in Figure 5.23) indicates the number of relationships the element currently participates in. Qualitative practitioner feedback reported in [RHM13b] strongly suggests that this counter is meaningless in practice. However, given the empirical basis it is too soon to draw any final conclusions.

Existing relationships already defined in the EA repository can also be manipulated. Thereby, the visualization supports removing a single relation (⑨ in Figure 5.23), i.e. APPLICATION COMPONENT or NODE DEVICE, or the entire relationship (⑩ in Figure 5.23), an INFRASTRUCTURE INTERFACE instance in our example. Using this visualization, various end-users are able to resolve conflicts and are supported with additional validation techniques to achieve this goal. These modifications in the conflict resolution visualization are propagated to the EA repository. Note that the presented concept abstracts from concrete types such that any model elements could be mapped, and, thus, any abstraction gap within an arbitrary EA metamodel could be harmonized [HMR⁺12, HRP⁺13b, RHM13b, SMR12].

The Abstraction Gap Resolver is seamlessly integrated in the task as it is part of its configuration. It can be used during the conflict resolution process (cf. Section 5.2.7) as the task can be forwarded and the state of the interactive visualization is based on the EA model, i.e. allows to collaborate asynchronously by forwarding the task with its configuration.

5.2.7 Resolve Conflicts

In [RHM13b], we introduce a conflict resolution process. Figure 5.24 depicts this iterative process. Its main success scenario starts with a conflict set that can be considered a Conflict Information Container (CIC) of a task and contains a description of the conflict. This set includes conflicts that could not be resolved automatically during importing and merging of models.

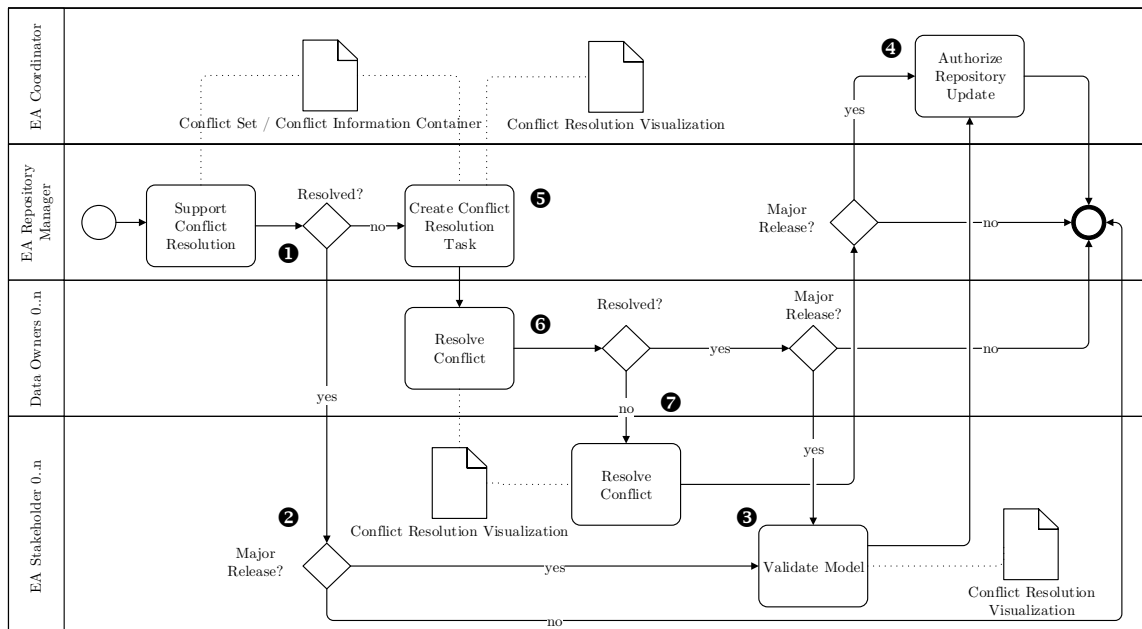


Figure 5.24: Conflict resolution sub-process [RHM13b]

The first escalation instance is the EA repository manager. Since the EA repository manager issues the merge process, it is the role's obligation to support the conflict resolution and to try to resolve arising conflicts (cf. Section 4.1.1.1 and ❶ in Figure 5.24). Conflicts that can be resolved by the EA repository manager are either forwarded for further validation or directly applied to the respective models (cf. ❷ in Figure 5.24). Further validation (cf. ❸ in Figure 5.24) is required in case of major releases⁵. In such a case, an EA stakeholder first validates the model and subsequently, the EA coordinator authorizes the update (cf. ❹ in Figure 5.24).

Note that tasks embrace descriptions of conflicts, which have been detected automatically (cf. Section 5.2.5), and conflicts, which have been detected manually by either the EA repository manager or an EA modeling expert (cf. Section 5.2.6). The conflict tasks (cf. Section 5.2.5.5) serve to validate performed actions. Specific tasks lead either directly to the conflict management dashboard (cf. Section 5.3.2) or to a special purpose visualization (cf. Section 5.2.6.2) that serves to understand and to resolve a conflict. While the former incorporates all task types, the latter is configured by the EA repository manager who is unable to resolve the conflict (cf. ❶ in Figure 5.24). Thus, the EA repository manager creates such a task manually with the respective configuration that describe the conflict (❺ in Figure 5.24). In this step, conflict-affected entities to be resolved are selected and the task is delegated to the responsible role of the `MODELELEMENT`. By default, the addressee of the task is the responsible role of an import model, i.e. a data owner of an information source. In our main success scenario, we assume that the responsible data owner receives the newly created task including the conflict resolution visualization (cf. Section 5.2.6.2) in a worklist (cf. Figure 5.21 on p. 182). The data owner is responsible for the information source and, thus, is able to resolve most conflicts without involving an EA stakeholder (❻ in Figure 5.24).

When modifying an element with an OID that incorporates a surrogate key to an external system, i.e. an information system of an information source, a propagate task is generated. This generally holds true for all modifications of `MODELELEMENTS`, not only for this special configuration. The core idea is that the propagate task contains all relevant information such that a modification on an import model can be understood by a human actor and reproduced within the original information source (cf. also Section 5.2.5.5). As stated in Assumption 4.7, we establish that CCMC is an aspired vision the community member constantly share. Facing reality, CCMC will stay such a vision yet to be achieve by collaboration between its individual members. The propagate tasks serve as a means to work toward this vision. We define CCMC as follows:

⁵We refer the interested reader to [RHM13b] for feedback from industry experts and a discussion about the criteria of major and minor updates of an EA model.

DEFINITION 5.7: Cross-community model consistency

In a federation with $\mathcal{M}_{2..n} \xrightarrow{\circ} \mathcal{M}_1$ cross-community model consistency is achieved iff:

1. shared OBJECTS conform to their OBJECTDEFINITION defined in \mathcal{M}_1
2. ATTRIBUTES conform to their ATTRIBUTEDEFINITION defined in \mathcal{M}_1
3. type constraints defined in \mathcal{M}_1 are not violated
4. cardinality constraints defined in \mathcal{M}_1 are not violated
5. queries to elements of \mathcal{M}_1 return the same (semantic) results as queries to respective elements in $\mathcal{M}_{q_1..q_n}$ whereas $\mathcal{M}_{q_1..q_n} \subseteq \mathcal{M}_{2..n}$ are considered the sources of the queried information.

■

If the data owner resolves the conflict and respective modifications are regarded as a major update, the validation by EA stakeholders and subsequent authorization by the EA coordinator (cf. ③ and ④ in Figure 5.24). Otherwise, the changes are immediately applied to the involved models.

On the other hand, if the data owner is unable to resolve the conflict, the task is forwarded to an EA stakeholder (⑦ in Figure 5.24). That means only if the EA repository manager and the data owner are unable to resolve the conflict, the respective EA stakeholder needs to be contacted. In this case, the EA stakeholder receives the task describing the conflict. At this point, the task might be annotated with comments from the data owner or the EA repository manager. This additional information allows to collaborate asynchronously. We argue that the EA stakeholder might not have any technical background [KW07], sufficient skills, access rights, etc. to maintain the EA tool. In [RHM13b], we provide qualitative feedback from EA experts indicating that the provision of visual means increases utility in the course of conflict resolution. Since EA Stakeholders are used to interact with EA information visually (cf. [MBL⁺08, RZM14]), we regard interactive visualizations as an intuitive way to empowered EA Stakeholders to resolve conflicts.

If a particular EA stakeholder is not in possession of the knowledge required to resolve the conflict, the task might get delegated to other stakeholders. Resolved conflicts are either directly propagated to the EA repository or the EA coordinator is asked to authorize the update if a major release is issued (④ in Figure 5.24).

The proposed process realizes an escalation of the model conflict and only involves EA Stakeholders if the EA repository manager and the data owners are unable to resolve conflicts. This escalating behavior can be represented in MODELGLUE with the chain of responsibility described in (cf. Section 5.1.2.4).

Note that not all practitioners agree with the final approval step denoted ‘Authorize Repository Update’ in Figure 5.24. However, practitioners agree that this process has a highly iterative and collaborative nature. Further, we found that practitioners value visual means to resolve conflicts or report the status-quo as a basis for discussion [RHM13b].

5.2.7.1 Learning and Batch Processing

In his bachelor’s thesis [Sc13], Schrade proposes a mechanism to detect frequently applied users choices throughout the conflict resolution process. The mechanism ‘learns’ from previous decisions and proposes the user to apply similar solutions to all remaining conflicts. Thereby, he distinguishes between the dimensions

Model: Are chosen changes part of a particular model?

Role: Are chosen changes issues by a particular role?

Time: Within which of the clusters {latest, oldest, between} are the chosen changes?

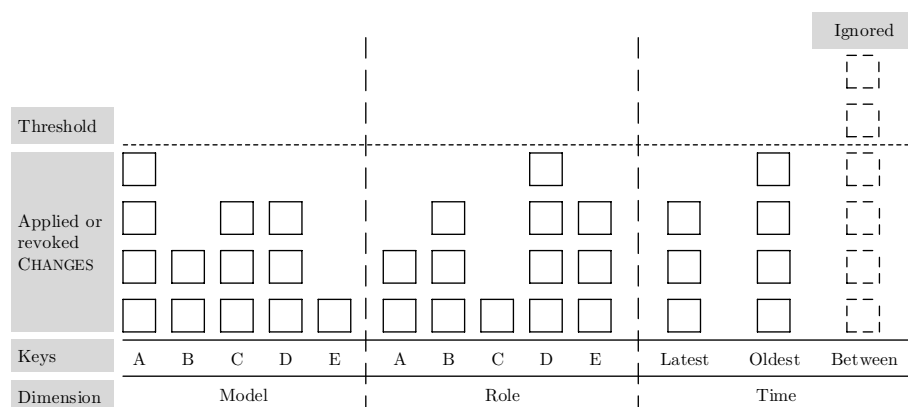


Figure 5.25: Learning mechanism as proposed by Schrade

Figure 5.25 illustrates this mechanism. It depicts the different dimensions observed by the approach and their keys. For each applied CHANGE, the system checks whether the CHANGE falls in such a dimension. The keys depend on the observed dimension; for models and roles, the key is determined based on meta-information of an applied change, i.e. if that change originates from a specific model or has been issued by a certain role. For time, MODELGLUE checks if a change is issued in a certain range of time that has been calculated based on an analysis of the clusters {latest, oldest, between}.

For each key, a certain threshold exists which depends on the number of conflicts to be resolved. The idea of our design is that for many conflicts, a fixed upper bound exists whereas for few conflicts, the learning mechanism is not used at all. In [Sc13, p. 41ff], Schrade details how these fixed values are calculated. Informally, there exist lower and upper bounds for which absolute values are used whereas for any other amount, a threshold relative to the number of tasks to resolve by a user is calculated.

Figure 5.26 depicts the UI part of the learning mechanism. Besides agreeing or disagreeing to proceed with the recommended way to solve conflicts, the user can choose to not being asked again by the system. This disables the active learning mechanism (cf. ‘ignored’ in Figure 5.25). However, the usage statistics can still be applied in a different context.

This learning mechanism is part of a three-fold approach to ease and automate conflict resolution. While this mechanism supports during a conflict resolution session, the states

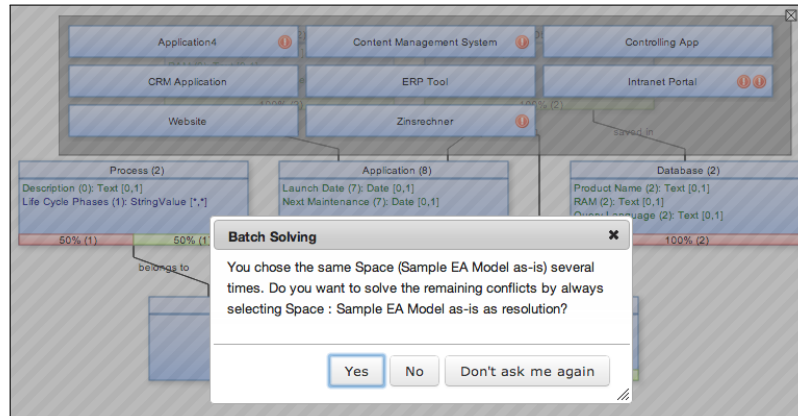


Figure 5.26: Confirmation dialog developed by Schrade [Sc13] and as illustrated by Kirschner in [Ki14, p. 47]

of a task are a way to prevent recurring inconsistency-reports that would only contain information one already knows about but decided to keep these inconsistencies (contrary to the idea of CCMC). Although this learning mechanism serves users as a support during a single conflict resolution session and does not have a direct impact on the applied conflict resolution strategy, it can serve an identification of recurring patterns that are applied in the course of conflict resolutions. Once identified, information about the applied pattern can serve as input for our third mechanism to automate the resolution of conflicts: the adaptation of the conflict resolution strategy which is discussed in the next section.

5.2.8 Adaptable Conflict Resolution Strategies

Storing operations as CHANGESETS, we can compare different CHANGES in CHANGESETS of the models to be merged in order to detect conflicts in the course of a model merge. Although we motivated that most model conflicts require manual intervention, we outline two different strategies that can be applied to merge models in a semi-automated fashion, i.e. different models are merged and some additional work is delegated to responsible roles to resolve arising conflicts. These are: strict and *tolerant* conflict resolution strategies. Especially the latter strategy offers a considerable degree of freedom since it merges elements of the metamodel and model without enforcing conformance of the model to its metamodel. Figure 5.27 illustrates the relationship of these strategies and which tasks for conflicting events it defines.

We first detail a strict strategy to detect conflicts in Section 5.2.8.1. This strategy avoids any unrecognized changes during a merge, requires approvals on deletions, and informs about each and every CHANGE. Further, we introduce a less strict variant in Section 5.2.8.2. This strategy represents the minimal conflicts without further knowledge of the semantics of a model and its metamodel. As illustrated in Figure 5.27, we allow users to define their own actions to cope with specific conflicts. This way, conflict resolution strategies can incorporate expert knowledge and can be adapted for each organization individually. This mechanism is detailed in Section 5.2.8.4.

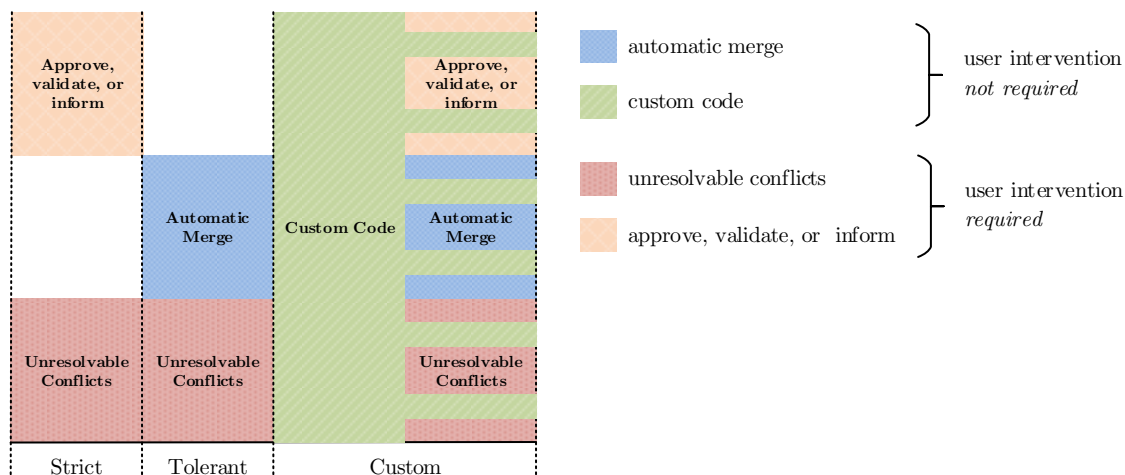


Figure 5.27: Overview of conflict resolution strategies

5.2.8.1 Strict Conflict Resolution

The following principles have been incorporated in the strict conflict resolution strategy:

- inform users about any concurrent update of related MODEL_ELEMENTS,
- require approvals for deletions of changed MODEL_ELEMENTS,
- ensure that all automatically merged data conforms to a definition,
- validate that no additional constraint violations occur after a merge, and
- do not merge metamodel/metamodel conflicts automatically.

That means the strict conflict resolution strategy generates TASKS as soon as a responsible role could be informed about CHANGES that might not be reviewed by this role. All information of the merged models is retained in generated TASKS such that effectively no data will be lost after a merge of models. However, the latest information might not be incorporated in the EA model, because it only is send to the responsible ROLES and needs further human intervention. Any transaction that is performed by this conflict resolution strategy is recorded in the version history and can be reverted if necessary.

We refer the interested reader to Kirschner [Ki14, Appendix A]. In his master's thesis, he details this strategy. The strict strategy is pessimistic, i.e. nothing can go wrong when it is applied and constraint violations remain the same after a merge of models. However, this strategy requires considerably more user intervention than the tolerant conflict resolution strategy.

5.2.8.2 Tolerant Conflict Resolution

The tolerant conflict resolution strategy guarantees that tasks are generated only if inevitable. Similar to the strict conflict resolution strategy, any information is kept by the system such

that none will be lost. In contrast to the strict conflict resolution, the tolerant conflict resolution applies more information to the target model. For instance delete operations are immediately performed without the need of an approval. The tolerant conflict resolution strategy also does not inform about missed updates; however, all transactions are recorded in the version history and can be reviewed and—if necessary—reverted to the previous state. The tolerant conflict resolution seeks to merge all data in the target model such that additional constraint violations are more likely to occur.

The tolerant conflict resolution strategy

- merges elements of the metamodel and model without enforcing conformance of the model to its metamodel
- does not inform users about concurrent updates on related MODEL_ELEMENTS, and
- does store all information to track CHANGES.

We refer the interested reader to Kirschner [Ki14, Appendix A]. In his master’s thesis, he details this strategy.

5.2.8.3 Merging Constraints

The tolerant strategy exploits the advantage of a loose coupling between metamodel and model that is realized by the metamodel introduced in Section 5.1.2. In the following, we briefly outline how constraints can be merged within MODELGLUE.

5.2.8.3.1 Cardinality Constraints

Informally, two cardinality constraints are equal to each other if their upper and lower bounds are the same. Unequal cardinalities must be merged based on made changes.

Table 5.3 depicts the different actions taken by MODELGLUE given two different CHANGES that intend to alter constraints of an ATTRIBUTEDEFINITION. The cardinalities are either merged without notification (tolerant) or a user must choose an option. After the merge transaction is completed, the system subsequently decides whether to check consistency of the involved model elements. Elements involved are thereby given through the relationship of the respective CHANGESSET, i.e. for two given CHANGES δ_1, δ_2 we check elements for which $\delta_1.changeset \hat{\vee} \delta_2.changeset$ holds. Commonly that means to check all attributes in all objects which maintain that attribute; a CONSTRAINTVIOLATION occurs if they do not conform to their definition.

As depicted in Table 5.3, the tolerant conflict resolution strategy can always resolve such conflicts by transforming the cardinalities to a many-to-many relationship denoted $n..m$. The entry $\hat{\vee}$ in Table 5.3 thereby refers to a check whether an object with $n..1$, $1..n$, $0..n$, or $n..0$ exists. If for instance, two changes of a cardinality constraint with $n..1$ and $1..n$ exist, all OBJECTS are analyzed for the actual frequency an attribute occurs. If no object features $n..1$ and the user chose to apply $1..n$ instead, CONSTRAINTVIOLATIONS are created to

5. Federated EA Model Management Design

$\delta_1 \backslash \delta_2$	0..n		1..n		n..m		n..1		n..0	
	strict	tolerant	strict	tolerant	strict	tolerant	strict	tolerant	strict	tolerant
0..n	—	—	$\hat{\oplus}$ if δ_2	δ_2	$\hat{\oplus}$ if δ_1	δ_2	$\hat{\oplus}$ if $\delta_1 \vee \delta_2$	$n..m$	$\hat{\oplus}$ if $\delta_1 \vee \delta_2$	$n..m$
1..n			—	—	$\hat{\oplus}$ if δ_2	δ_2	$\hat{\oplus}$ if $\delta_1 \vee \delta_2$	$n..m$	$\hat{\oplus}$ if $\delta_1 \vee \delta_2$	$n..m$
n..m					—	—	$\hat{\oplus}$ if δ_2	δ_1	$\hat{\oplus}$ if δ_2	δ_1
n..1							—	—	$\hat{\oplus}$ if δ_1	δ_2
n..0									—	—

$\hat{\oplus}$ consistency-check after applying the change

Table 5.3: Merging cardinality constraints

report this deviation between the model (OBJECTS with ATTRIBUTES) and its metamodel (ATTRIBUTEDEFINITION). On the other hand, if changes that alter cardinality constraints by $n..m$ and $1..n$ are issued and one chooses to apply the first change, i.e. $n..m$ cardinalities, federal consistency—with respect to this operation—is guaranteed such that no further checks have to be performed.

An alternative design of the tolerant merge strategy is to first check if CONSTRAINTVIOLATIONS occur and afterwards use the most rigid cardinalities for which at the same time the fewest CONSTRAINTVIOLATIONS occur. This way, it is guaranteed that the most specific cardinalities ($n..1$, $1..n$, $n..0$, $0..n$, or $n..m$) are chosen to merge information in a tolerant manner.

5.2.8.3.2 Optionality

Whether the optionality of an ATTRIBUTEDEFINITION is changed during a merge depends on the kind of MODELELEMENTS merged. Since deviations of an OBJECT to its OBJECTDEFINITION are tracked as CONSTRAINTVIOLATIONS, optionality is only changed if at least two CHANGES on congruent ATTRIBUTEDEFINITIONS have to be merged. If two transient CHANGES seek to modify an optionality constraint of an ATTRIBUTEDEFINITION, one CHANGE with `optional` and the other with `mandatory`, the tolerant conflict resolution strategy of MODELGLUE modifies the ATTRIBUTEDEFINITION such that the optionality is always set to `optional`. This way, CONSTRAINTVIOLATIONS are avoided and information is accessible for humans. The strict conflict resolution strategy on the other hand does not merge optionality and creates a resolve task (cf. Section 5.2.5.5).

5.2.8.3.3 Type Constraints

In line with Schweda [Sc11, p. 181], we take into account that objects of the types $\{byte, integer, decimal, boolean, date, time, dateTime, duration, URI, \dots\}$ (cf. also [ISO07a, p. 28ff]) have a lexical representation [W304b], i.e. they can be converted to a string representing the value as literal. For our next considerations we introduce the notion of rigidity between types.

DEFINITION 5.8: Type order

If \mathcal{T}_a can be subsumed by \mathcal{T}_b , we write $\mathcal{T}_a \gg \mathcal{T}_b$ which reads as \mathcal{T}_b is less rigid than \mathcal{T}_a . ■

The tolerant conflict resolution strategy follows the assumption that the more rigid the type is, the more constraint violations are to expect. Thus, the optimistic, i.e. tolerant, strategy always uses the less rigid type. For source type \mathcal{T}_s and target type \mathcal{T}_t the merged type \mathcal{T}_m is given by Equation 5.24.

$$\mathcal{T}_m = \begin{cases} \mathcal{T}_t & \text{if } \mathcal{T}_s \gg \mathcal{T}_t \\ \mathcal{T}_s & \text{else } \mathcal{T}_t \gg \mathcal{T}_s \vee \mathcal{T}_s \equiv \mathcal{T}_t \end{cases} \quad (5.24)$$

We assume an Abstract Rewriting System (ARS) for types as defined in Equation 5.25.

$$\begin{array}{ll} \text{Text} & \gg \text{Long text} \\ \text{Number} & \gg \text{Text} \\ \text{Date} & \gg \text{Number} \\ \text{Enumeration} & \gg \text{Number} \\ \text{Image} & \gg \text{Text} \\ \text{Reference} & \gg \text{Text} \end{array} \quad (5.25)$$

Note that images and references can be represented as text encoding their URI. Further, one must take into account a possible precision loss when converting the type date. This not only embraces the exact format but also its precision, e.g. is it stored as a timestamp in milliseconds or seconds, as well as its timezone. Similar problems arise from number to text conversation as the mantissa may play an important role interpreting the number.

5.2.8.4 Adapting the Conflict Resolution Strategy

Besides using one of the predefined conflict resolution strategies, MODELGLUE allows to define an organization-specific conflict resolution strategy. More precisely it allows to adapt the current conflict resolution strategy. Thereby, a matrix can be used to select whether a predefined strategy or a customized operation should be executed.

ASSUMPTION 5.1: Customized conflict resolution strategy

We assume that a customized conflict operation resolves a conflict, i.e. conflicting changes are either accepted, discarded, or other actions are performed which resolve the conflict.

5.2.8.4.1 Customizing an Existing Strategy

Organizations may want to deal with some conflicts in a specific way. For instance changes, issued from one role, might be preferred over others. In our evaluation, practitioner report that information stored in a COTS product is not that ‘reliable’ than in a custom developed application (cf. Chapter 7).

5. Federated EA Model Management Design

As illustrated in Figure 5.27 on p. 190, custom code can be executed in any conflict situation instead of using one of the predefined operations that are used in the strict and tolerant merge strategies. We assume that by executing the customized code, the conflict is resolved.

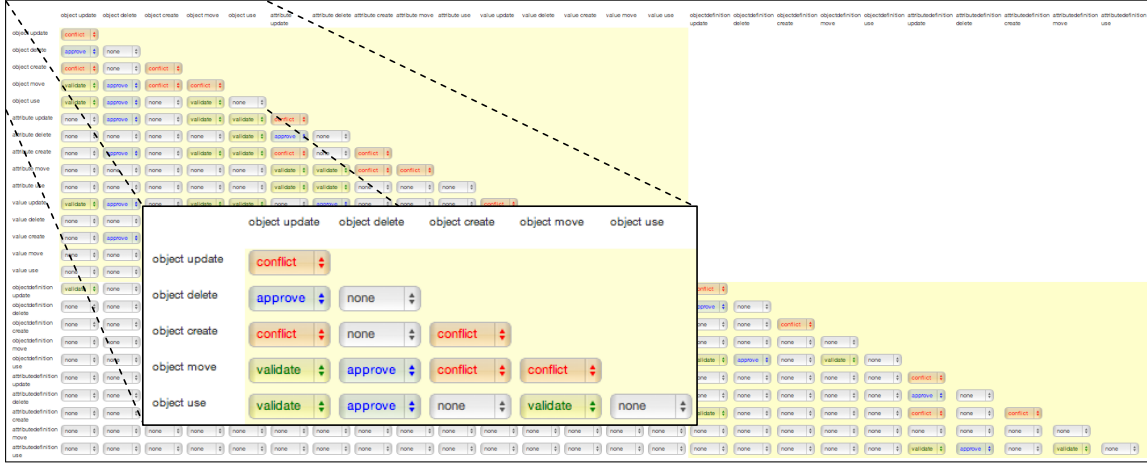


Figure 5.28: Conflict resolution strategy widget

Figure 5.28 shows the conflict resolution strategy. Note that we provide two pre-defined strict and tolerant strategies (cf. Section 5.2.8.1 and Section 5.2.8.2) such that organizations are only tasked to refine the matrix rather than to configure it from scratch. An organization can adapt the conflict resolution strategy to individual needs. Customized rules could also trigger customized human tasks which implement further application logic; however, in the current design the underlying language does not support to create tasks. The matrix is a LUT for $\text{MODELELEMENT } \mathcal{E}_1, \mathcal{E}_2$ and $\text{CHANGES } \delta_1, \delta_2$, i.e. $\text{getType}(\mathcal{E}_1) \times \delta_1.\text{operation} \times \text{getType}(\mathcal{E}_2) \times \delta_2.\text{operation}$ whereas $\text{getType}()$ is a utility function to determine the classifier of the subclass of a MODELELEMENT .

For each cell, users can choose to configure whether they want to raise a specific task type or invoke a customized rule. Cells that are configured with a predefined task type must at least adhere to the tolerant conflict resolution strategy. However, this does not hold true for customized rules since we assume that by executing the customized rule, the conflict is resolved. Equation 5.26 formalizes this lookup.

$$rule_i^j = \begin{cases} tolerant_i^j & \text{if } user\ choice \notin tolerant_i^j \wedge \nexists customRule_i^j \\ customRule_i^j & \text{else} \end{cases} \quad (5.26)$$

5.2.8.4.2 User Interface Support for the Definition of Custom Merge Strategies

The customized rules which can be incorporated in a merge strategy can be specified by users. This way, users can implement their own rules that resolve a conflicting situation. Inspired by the user interface of iTunes, we develop a dialog for specifying these structured merge rules. Before we continue with the illustration and description of our design, we revisit the principles from the iTunes UI we adapted. Figure 5.29 illustrates the dialog to

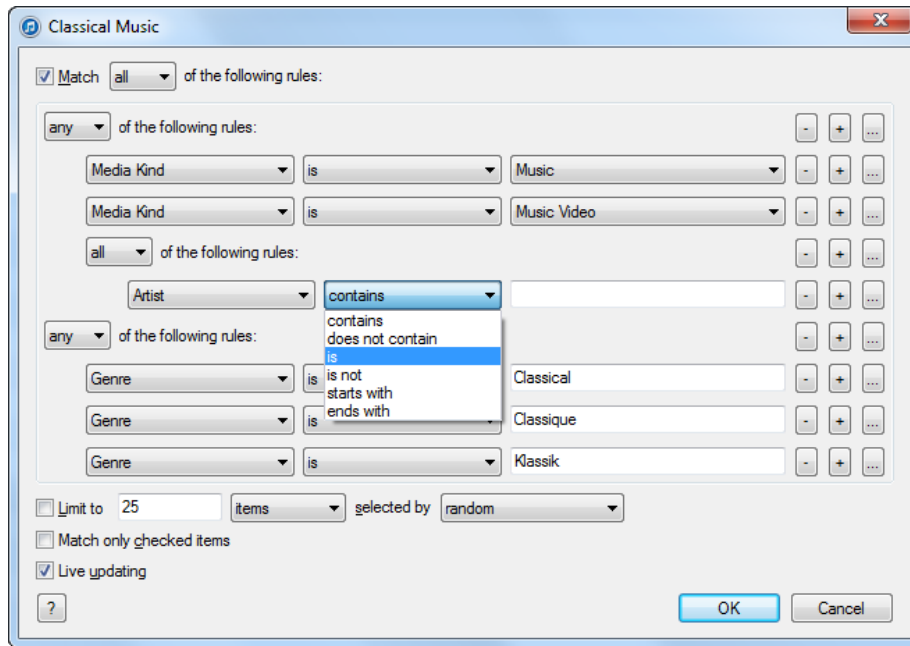


Figure 5.29: Filter to customize playlists in iTunes

customize playlists in iTunes. The user is able to specify different filters using an intuitive interface to specify rules that either match to all elements or any element in a music collection. Besides this quantification, users may choose from a list of attributes of a music file, here: media kind, artist, and genre. Depending on this type, operations to compare are given. These differ for strings (starts with), categories (is/is not), or numbers such as ratings (less than, more than). User concatenate statements to the collection that apply to **all** or **any** conditions.

MODELGLUE incorporates a similar design to specify merge rules with model expression language (MxL) [Re13]. While the iTunes dialog serves users to specify a filter that is applied to a collection of elements, our solution intends to specify rules applied to a specific conflict.

Kirschner [Ki14, p. 46] presents a design to specify multiple merge rules at once. Figure 5.30 illustrates a dialog⁶ to configure rule-based actions that are invoked in the event of particular model conflicts. The general structure consists of two boolean expressions (① and ② in Figure 5.30) which check conditions of involved MODELELEMENTS and may trigger an action (③ in Figure 5.30). Each condition can be restricted to a particular type kind of MODELELEMENT (④ in Figure 5.30) which can be chosen from a drop-down list (⑤ in Figure 5.30). This design brings together the notion of a filter (cf. also Section 5.3.1.5 and Section 5.3.2.6) and the definition of rules for entire ranges of operations. As illustrated, new conditions for an element can be added or removed (⑥ in Figure 5.30) which sets the scope of the rule beyond the possibilities of type checking of MODELELEMENTS. These conditions are evaluated regarding the applied change, i.e. the comparison assumes that the change already is applied to the MODELELEMENT. This can be accomplished since both

⁶Note that ‘*object*’ reads as MODELELEMENT and ‘*Page*’ as OBJECT in Figure 5.30.

MODELELEMENTS are physically in one system and represented by a persistent version, i.e. the latest revision of a MODELELEMENT in different branches. This way, the user compares states. Note that reaching this rule already implies that both MODELELEMENTS are related (cf. Section 5.2.5.1). An action on the other hand (7 in Figure 5.30) can be one of {update object, delete object, create new Object, ...}. We envision situations in which attributes must be updated, deleted, or created (8 and 9 in Figure 5.30).

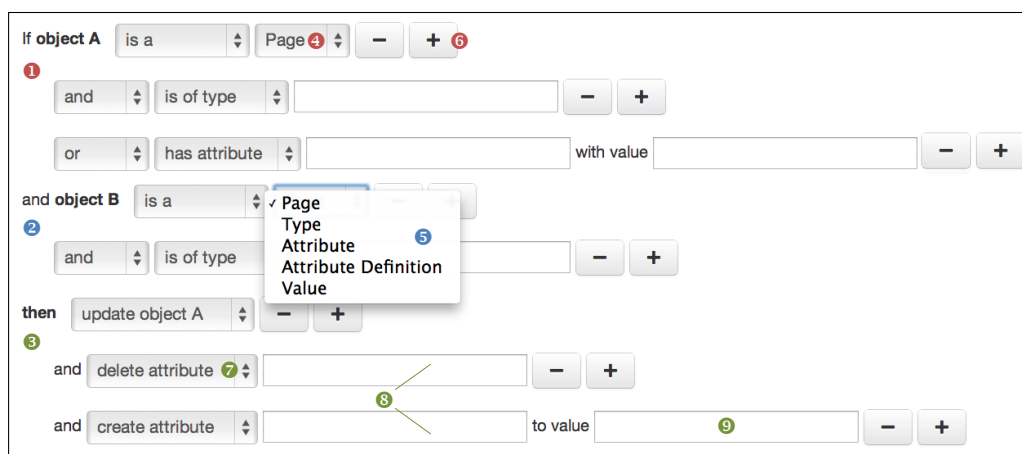


Figure 5.30: Annotated screenshot of the dialog to specify merge rules developed by Kirschner [Ki14, p. 46])

An alternative design of this dialog has a smaller focus, is more precisely in its context, but requires considerably more effort to define rules for entire ranges of operations or an entire conflict resolution strategy. Since our idea is that an organization adapts a predefined resolution strategy in an incremental manner, we considered an alternative UI shown in Figure 5.31. At a first glance, it conveys a lean design. It has been our intent to initially hide complexity from the user.

In contrast to the design proposed by Kirschner, this dialog already specifies two concrete types of a MODELELEMENT (1 in Figure 5.31) and two concrete OPERATIONS concurrent CHANGES intend to apply in the course of the merge (2 in Figure 5.31), i.e. a rule specified by this dialog represents one cell in the conflict resolution strategy matrix (cf. Figure 5.28). Commonly, for one of Kirschner's range-rules, one has to configure one rule for each operation issued for involved objects. To recap what was detailed in Section 5.2.5.6, one could end up with 25 case for each range-rule. However, we foresee that Kirschner's design bares the risk of over-generalizing cases. Range-rules have drawbacks as they only compare states of objects with no context information about the performed operation whatsoever. This makes it considerably harder to cope with involved complexity once more rules, e.g. on cell level of the conflict resolution strategy, are involved.

We kept the basic if-then structure. At the bottom, an action can be defined (3 in Figure 5.31) that is applied on a query result which must be a MODELELEMENT (4 in Figure 5.31). Depending on the chosen operation, another query is shown (5 in Figure 5.31). In this case, a query for a MODELELEMENT is set to another query result which is composed of attributes of both changed MODELELEMENTS. If both arguments of the '+' operator are strings, MxL detects the operation as a concatenation of strings [Re13, p. 27] and returns

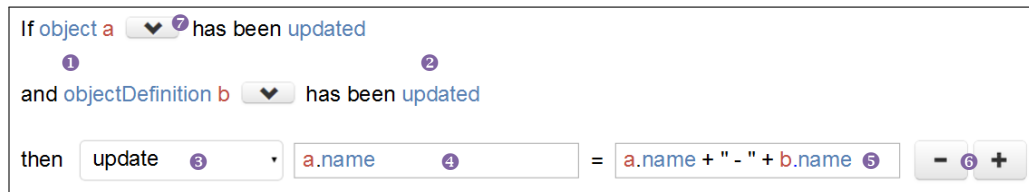


Figure 5.31: Screenshot of the dialog to specify merge rules (minified)

one string. Note that elements **a**, and **b** are injected in the expressions scope and are, thus, accessible for the user via the identifier **a**, and **b**. If required, the dialog can be expanded in two ways to support more complex operations. First, multiple actions are supported (6 in Figure 5.31) as well as the scope of *both* involved MODELELEMENTS can be specified by adding conditions (7 in Figure 5.31).

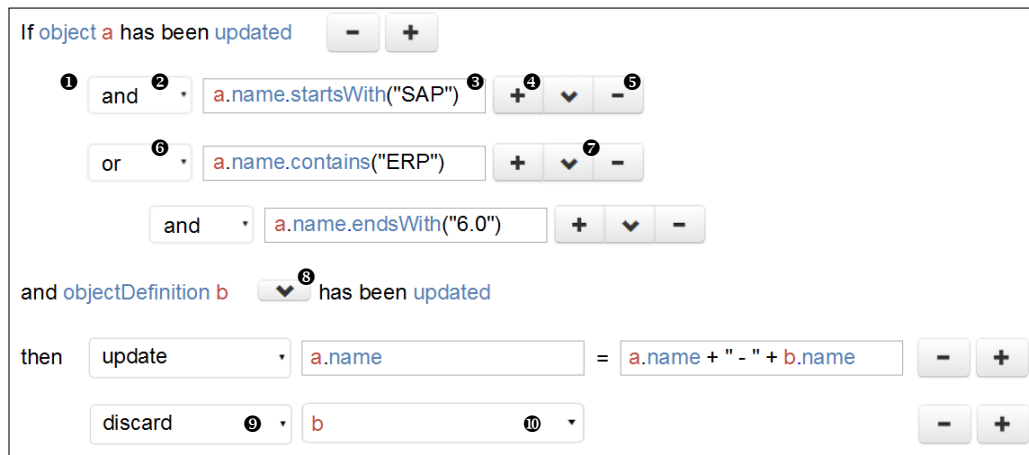


Figure 5.32: Screenshot of the dialog to specify merge rules (expanded)

On click, a condition is added (1 in Figure 5.32). Thereby, the first condition always comes with the logical **and** conjunction (2 in Figure 5.32). A condition is a boolean expression specified in MxL (3 in Figure 5.32). Further conditions can be either added or removed (4 and 5 in Figure 5.32). Conjunctions only embrace **and** and **or** (6 in Figure 5.32) since MxL implements the actual logic and also includes **not**. Via indenting conditions (7 in Figure 5.32) additional criteria can be specified. Conditions can be specified for both MODELELEMENTS involved in a conflict (8 in Figure 5.32) as well as one can specify further actions (9 in Figure 5.32). In this example, changes on **b** are discarded (10 in Figure 5.32). We also foresee that performing consistency checks and other mechanisms to ensure model quality could be performed. Since this dialog can define queries for any MODELELEMENT and its ATTRIBUTES accessible by MxL, further build-in system properties such as ‘last-modification date’, ‘creation date’, ‘responsible role’, ‘last writer’, etc. are available to configure an organization-specific conflict resolution strategy.

Although the UI dialogs are meant to ease the configuration of an organization-specific conflict resolution strategy, we are aware that both UIs are considerable complex regarding the implied logic one is able to configure with simple click operations. However, both

designs aim at an experienced audience with advanced technical skills, i.e. EA repository managers or EA modeling experts.

5.3 Interactive Visual Support

Merging models can be regarded as a complex task. Such a complex task can be facilitated by advanced user interfaces to make use of human cognition that is especially strong in identifying patterns in complex information represented visually⁷. Visualizations are a common means in EA management not only to identify and prioritize problems but also to discuss the current state of an EA as well as to discuss planned states thereof in order to develop transition plans that implement changes.

In MODELGLUE, we employ advanced user interfaces that combine interactions known from UI components as well as common visualizations known from EA management. The interactive visual support discussed in this section is twofold. First, we propose visual support for analyzing differences in models and metamodels. Then, we introduce an *interactive conflict management dashboard*.

5.3.1 Visualizing Metamodel and Model Differences

MODELGLUE empowers users to view at the differences of two metamodels and differences of respective models visually. This view is generated with respect to the origin of both models. MODELGLUE represents model as well as metamodel differences in one interactive view augmented by additional information.

In [RM14], we present a preliminary version of an approach to visualize model and metamodel conflicts. The present thesis builds on this version and presents key concepts. Figure 5.33 illustrates a conceptual overview of the visualization. As suggested by Lidwell et al. [LHB10, p. 146], we use layering to manage complexity. The main idea is to use the metamodel as navigation to drill down to instances, i.e. OBJECTS and their ATTRIBUTES. That is, an OBJECTDEFINITION of the metamodel is color coded if any differences exist. The end-user can access information at four layers via different interactions. Thereby, filtering is essential as showing all the differences of two models (and their instances) is regarded as too complex and the result ends in an information overload (cf. [To70, SVV99]). Figure 5.33 illustrates the relationships of the four layers, the visual concept applied at each layer, and the typical number of instances we regard to interact with an EA model and respective metamodel. Visual objects at each layer serve the user as means to navigate. In the following we detail the different layers and give a visual example for each layer.

5.3.1.1 Layer 1: Metamodel Differences

In the context of coupled evolution, Krause et al. [KDG13] annotate an UML class diagram to illustrate their ideas. Thereby, the authors present the operations that were applied to

⁷For general design guidelines for visualizations we refer the interested reader to Fitts [Fi54], Tufte [Tu01], and Moody [Mo09]; for design guidelines for information dashboards we refer to Few [Fe06].

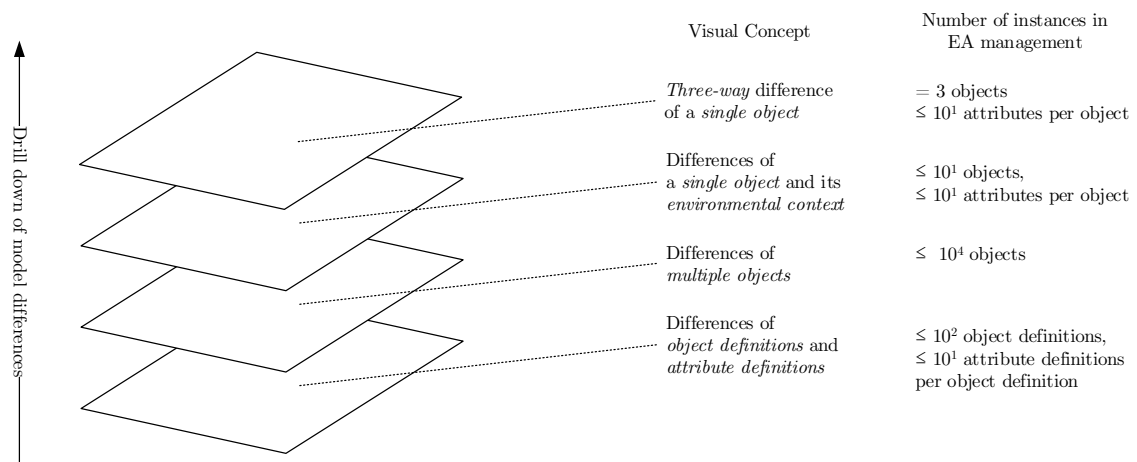


Figure 5.33: Four-layered interactive difference visualization for EA metamodels and models

different models visually. Krause et al. use gray color to encode concepts (classes) that are to be preserved in a model, red to highlight deleted concepts, and green for concepts that have been created. Besides color coding changes of constraints are added as an UML note to the diagram. We employ similar visual concepts.

In the first layer (see Figure 5.34), we visualize the differences of two meta-models denoted A and B. The entire metamodel is shown to the user as a graph. A graph layout algorithm is used to arrange the vertices and edges visual appealingly. An important design criteria to choose an algorithm is its resistance to change, i.e. the layout must be relatively stable with respect to small changes in the underlying data. A spring graph layout algorithm for instance commonly rearranges all the vertices such that a user ends-up with an entirely new layout each time the graph is rendered. New OBJECTDEFINITIONS are displayed in the background color green, altered classes in orange and deleted ones in red. In case the name of an OBJECTDEFINITION was altered, a two-way string diff (cf. [He78]) is applied to the respective name descriptors (see ‘Business Application’ in Figure 5.34).

ATTRIBUTEDEFINITIONS are displayed at this level as part of the OBJECTDEFINITION. The number of instances of every attribute, i.e. its actual usage, and the cardinality are given. New ATTRIBUTEDEFINITIONS are displayed green, deleted ones in red, and altered ATTRIBUTEDEFINITIONS are displayed 1) using a two-way textual differences algorithm, 2) showing version A and B, and 3) showing their origin. RELATIONSHIPS are illustrated as edges between the vertices in the graph. Newly created RELATIONSHIPS are displayed green, deleted ones red, and modified, i.e. the name, source, or target in orange. Each modification is annotated with the version a modification has been made. The respective versions of a modification (A, B, or A+B) is displayed in magenta colored text. This holds true for OBJECTDEFINITIONS, ATTRIBUTES as well as RELATIONSHIPS. Besides differences between OBJECTDEFINITIONS, their ATTRIBUTEDEFINITIONS and RELATIONSHIPS to each other, the first layer also displays aggregated information of model differences, i.e. the number of differences between OBJECTS of each OBJECTDEFINITION. Absolute and relative values of OBJECTS that have no differences (green, right part of the progress bar at the

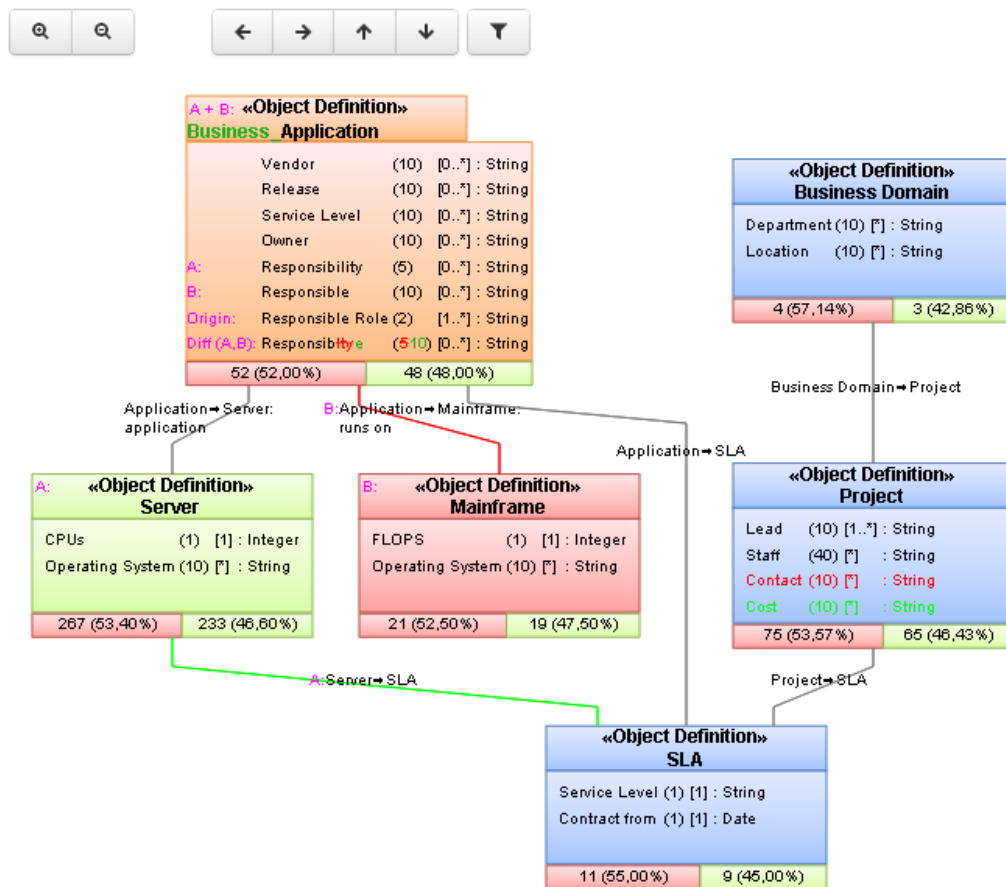


Figure 5.34: Layer 1: differences of two metamodels (schemas) A and B with aggregated information of differences in respective models (data)

bottom of each OBJECTDEFINITION), and OBJECTS that have differences in the other branch (red, left part of the progress bar of each OBJECTDEFINITION) are shown. This way, an end-user gets a good overview of model differences (per OBJECTDEFINITION) just by looking at the difference visualization of the meta-models. The user can either navigate to the respective OBJECTDEFINITION by clicking on its name or can click anywhere else on an OBJECTDEFINITION to open the next level: the instance overview for OBJECTS that conform to the clicked OBJECTDEFINITION.

We summarize the color coding of this layer as follows:

blue denotes an OBJECTDEFINITION is equivalent in both models,

black denotes an ATTRIBUTEDEFINITION or RELATIONSHIP is equivalent in both models,

red denotes an OBJECTDEFINITION, an ATTRIBUTEDEFINITION, or a RELATIONSHIP was deleted,

green denotes an OBJECTDEFINITION, an ATTRIBUTEDEFINITION, or a RELATIONSHIP was created, and

orange denotes the name of the OBJECTDEFINITION, a RELATIONSHIP, or at least one ATTRIBUTEDEFINITION thereof was changed in either of the compared models.

magenta denotes the respective model a change was made in.

Besides this color coding, a progress bar indicates the differences in the respective OBJECTS of an OBJECTDEFINITION, cf. Figure 5.34.

5.3.1.2 Layer 2: Model Differences Overview

To drill-down on respective model differences, a user clicks on an OBJECTDEFINITION which opens an overview of all instances. Again, the system shows differences using color codes similar to the differences between OBJECTDEFINITION and ATTRIBUTES.

blue means an OBJECT is equivalent in both models,

red means an OBJECT was deleted,

green means an OBJECT was created, and

orange means the name of an OBJECT or at least one ATTRIBUTE'S VALUE is different than in the model that it is compared to.

The user can click on an OBJECT to see its links to other OBJECTS including details of these OBJECTS. Figure 5.35 illustrates such a detailed view for the domain of EA management.

The second layer gives an overview of the model's instances, i.e. OBJECTS and ATTRIBUTES (cf. Figure 5.35). The end-user already narrowed the scope of the differences down to OBJECTS of a single OBJECTDEFINITION by the choice made in layer 1. As illustrated in Figure 5.33 on p. 199, the number of instances can grow considerably large in layer 2. For this reason, we add an intuitive filter to our interactive visualization. This way, users can define sophisticated range filters on multiple ATTRIBUTES (see Subsection 5.3.1.5). At this layer, our design goal was preventing information overload. We chose to hide any unnecessary details. As illustrated, just the name and the state of an OBJECT (via color coding) is shown to the user. The latter denotes whether an OBJECT was altered in *A*, *B* or both branches. When the user hovers over an OBJECT, its details are shown. Figure 5.35 further illustrates the controls available in any of the four layers. Besides basic facilities to zoom and navigate, the visualization can also be downloaded as Microsoft Powerpoint Format (PPT/PPTX) presentation for further manipulation.

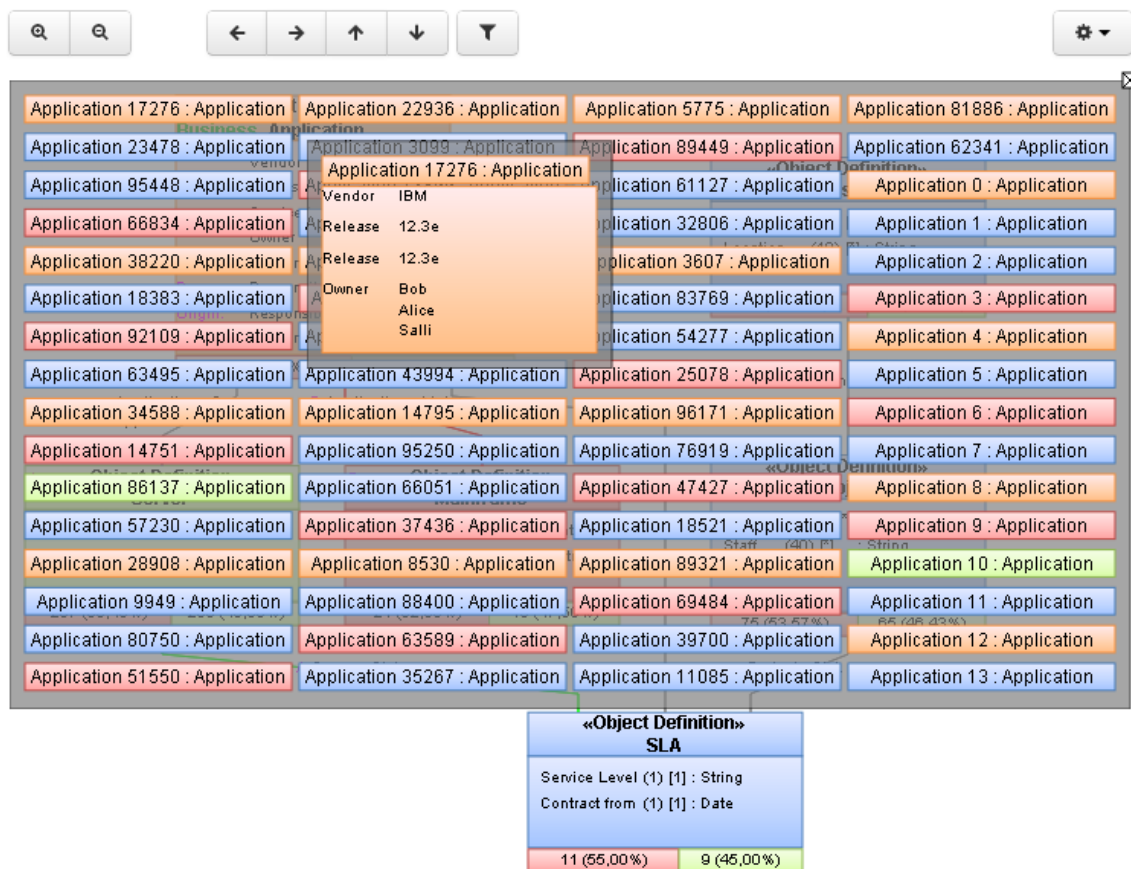


Figure 5.35: Layer 2: the instance overview with object details on mouse hover

5.3.1.3 Layer 3: Neighborhood Relationships

In EA management, not only the single OBJECT is of interest but the implications changes might mean considering the bigger picture, i.e. an OBJECT's environment. In layer 3, we show this environment to an end-user. Although EA management commonly analyzes aggregated information [WF06], more fine-grained information such as servers and routers also could be subject of analysis [BHS⁺12, BEG⁺12]. Considering to visualize all instances and relationships among them, displaying 10^8 visual objects at once might not be unrealistically. For this reason, we propose this consecutive drill-down on differences. The layout of the n^{th} -neighborhood of an OBJECT is again, arranged as a graph. Thereby, the depth of neighborhood OBJECTS to traverse (the n) is configurable. The same color coding and semantics like on the other layers are applied on this layer (orange: an OBJECT was modified in two branches; blue: no changes; red: OBJECT deleted, green: new OBJECT created). This layer picks up expert feedback on an earlier version (which is also a 'well-known' fact in EA management) that RELATIONSHIPS between OBJECT are far more interesting than changes of an ATTRIBUTE.

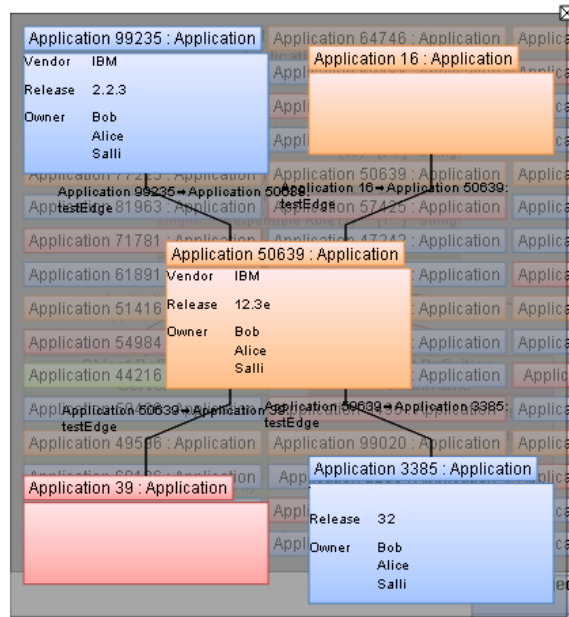


Figure 5.36: Layer 3: graph representation of an 1st-neighborhood of an instance

5.3.1.4 Layer 4: Three-way Difference

If and only if the object has differences, the user can again click on the object which opens another window showing a three-way comparison, i.e. both versions that have differences and, if existing, the common origin.

Another click brings the user to the last layer of the interactive visualization. At this layer, a three-way difference is shown, i.e. version A , B and (if existing) their origin (cf. Figure 5.37). Since only three instances are shown no further (string) differencing is needed.

We consider layer 4 the entrance point to navigate to the respective OBJECT within the system if any further details, such as access rights or modification date, are needed. Note that each of the layers 2 to 4 can be opened for multiple OBJECTS and the different views may overlap each other or switched by the user that analyzes the model differences.

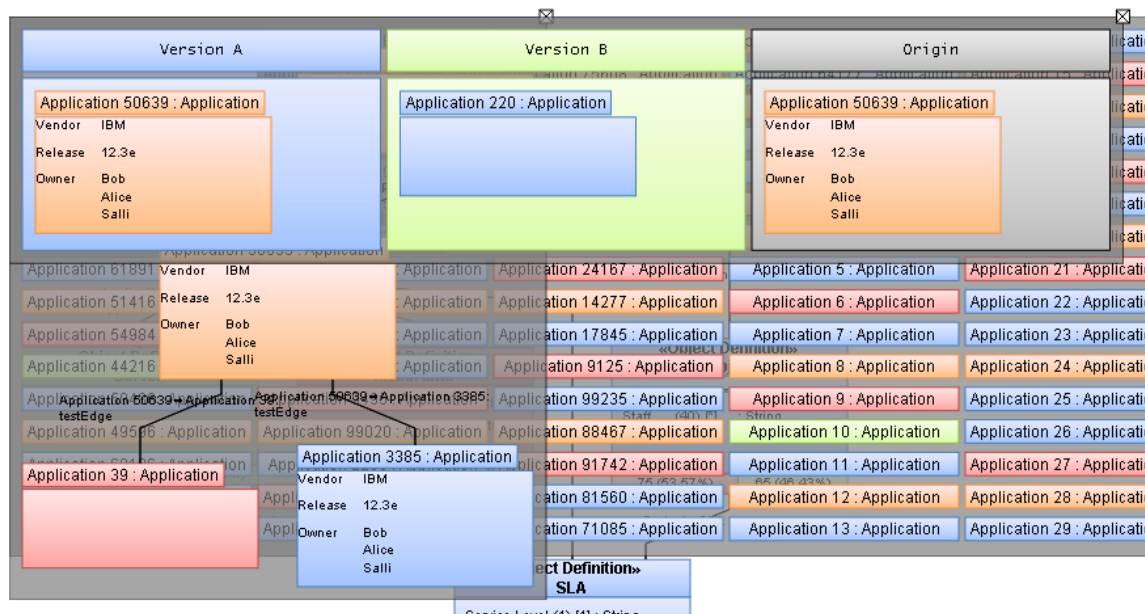


Figure 5.37: Layer 4: three-way difference view of an instance (top) and other layers (bottom/background)

5.3.1.5 Filtering

We facilitate the analysis of larger EA models, i.e. more instances, by employing a filter. A conceptual mock-up of the filter is depicted in Figure 5.38. It is meant to define a query for OBJECTS and can be applied in layer 1 and 2 preferably. The design was inspired by the iTunes playlist filter (cf. Figure 5.29 on p. 195). Our design goal was to empower EA experts to configure filters without the need to program or to adapt sophisticated search scripts. The filter can be applied to different OBJECTDEFINITIONS and, depending on the type of ATTRIBUTEDEFINITION that is subject to the filter, different comparators can be used. The user can specify different rules that are applied on a specific attribute of a type (cf. Figure 5.38). At the uppermost level, rules that are applied to a type are evaluated using a logical **or** and the logical **and**. At the same time, **or** and **and** can be applied to attributes of a type, too (cf. Figure 5.39). This way, multiple rules can be applied to an entire query as the granularity of a particular type.

The result of the filter dialog is a generated JavaScript Object Notation (JSON) string. The general structure of this string is sketched in Listing 5.1. In the prototypical implementation, this JSON string is processed as a server-sided query, i.e. it is applied to the result of the differencing algorithm. As illustrated, filters can be concatenated on the same level or nested recursively (cf. Figure 5.39 and Listing 5.1). This empowers EA experts to define sophisticated model queries.

Thereby, each attribute can be compared using the following comparators.

- String (applies to string values only)
 - contains, filters values for a substring \mathcal{S}_{sub} that contained in the string \mathcal{S}

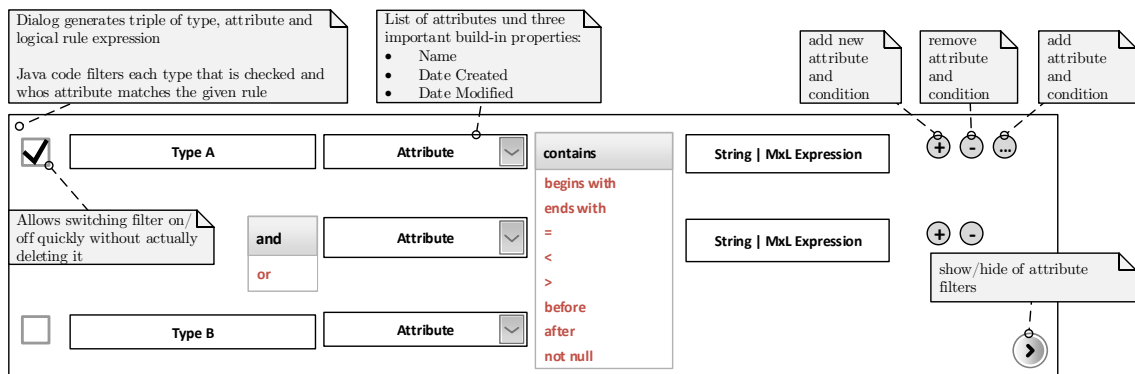


Figure 5.38: User interface design for a dialog to specify filter rules

- begins with, filters for a certain prefix of the string
- ends with, filters for a certain postfix of the string
- not null, filters any value that is not null
- equals, filters the string for a character-wise match with another string
- Number (applies to number values only)
 - operator =, filters values that match a given number exactly
 - operator >, filters values that are greater than a given number
 - operator \geq , filters values that are greater or equal than a given number
 - operator <, filters values that are less than a given number
 - operator \leq , filters values that less or equal than a given number
- Date (applies to date values only)
 - equals, filters values that are on an exact date and time
 - before, filters values that are before a given date
 - after, filters values that are after a given date

We consider this a non-exhaustive list of filter operations and others such as ‘equals ignore cases’ could be added. However, the more operations are added to the dialog, the more complex it gets for users to comprehend the dialog. As we seek for an intuitive design, together with EA practitioners, we concluded that above operations are sufficient for most queries. More sophisticated queries can be formulated using an expression language. Such a language on the other hand does require some understanding of programming or scripting languages. Albeit more expressive, we claim that using an expression language requires more learning efforts than employing an UI to define a query.

The general structure of the generated JSON string is as follows (cf. Listing 5.1):

- **Line 1:** a filter for one OBJECTDEFINITION,
- **Lines 2,7:** the possible conjunctions whereas the first conjunction defaults to **and**,

- **Lines 3,8:** the different kind of comparators users can choose from,
- **Lines 4,9:** a predicate that specifies the value which is evaluated against, and
- **Lines 5,10:** a tree of recursively nested filters that follow the same structure as the illustrated filters (lines 2–4, 7–9).

Listing 5.1: Schematic illustration of the JSON string produced by the filter dialog

```

1{"type" : [ {
2  "conjunction" : "and | or",
3  "comparator" : "contains | starts with | ends with | = | < | > | before | after
   | not null",
4  "predicate" : "value",
5  "filter" : [{"conjunction" : "...", "filter" : [{"conjunction" : "..."}]}]
6 }, {
7  "conjunction" : "and | or",
8  "comparator" : "contains | starts with | ends with | = | < | > | before | after
   | not null",
9  "predicate" : "value",
10 "filter" : [{"conjunction" : "...", "filter" : [{"conjunction" : "..."}]}]
11 } ]
12 }

```

We conclude this section by sketching a typical query that narrows the scope of an analysis to view model differences (see Example 5.6).



EXAMPLE 5.6: Filtering a difference visualization

The enterprise architect, John Doe, wants to view differences of the CMDB and the current EA model with a particular focus on servers that feature either 16 or 32 CPU cores. Since the latest CMDB update contained specific information on applications that run on these servers, John wants also to view the applications. To narrow the analysis, John specifies the amount of random access memory (RAM) for the servers; servers that have 16 cores should have at least 128 gigabyte (GB) RAM and servers with 32 cores at least 256 GB RAM. Figure 5.39 illustrates the resulting range filter.

<input checked="" type="checkbox"/>	Server	RAM in GB	≥	128	+ - ▾
	and ▾	Number of Cores	=	16	+ - ▾
	or ▾	RAM in GB	≥	256	+ - ▾
	and ▾	Number of Cores	=	32	+ - ▾
<input checked="" type="checkbox"/>	Business Application	Product Name	contains	String	+ - ▾
<input type="checkbox"/>	Projects	Owner	contains	String	+ - ▾
<input type="checkbox"/>	Costs	Name	contains	String	+ - ▾

Figure 5.39: Configured filter for the difference visualization



5.3.2 An Interactive and Collaborative Conflict Management Dashboard

In this section we elaborate on an interactive visualization that provides means to manage and resolve conflicts in a model collaboratively.

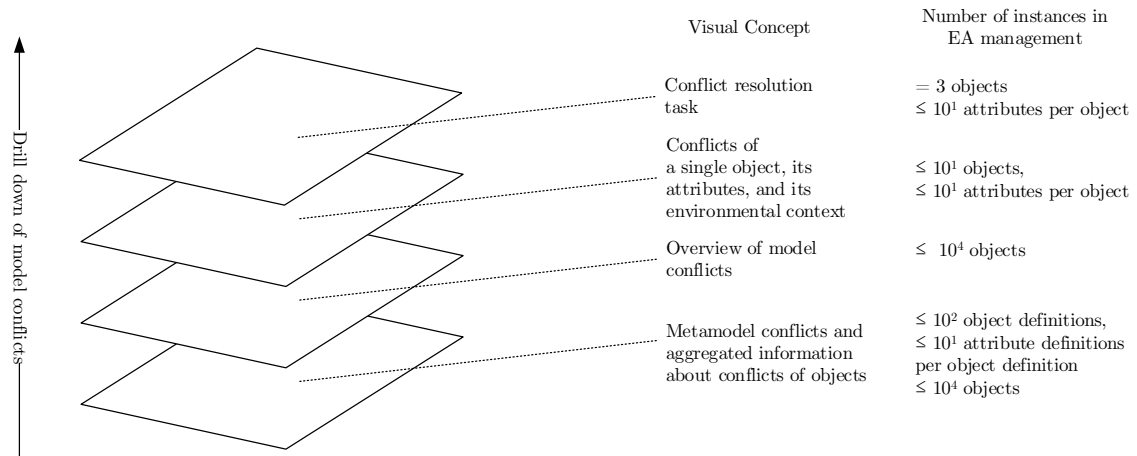


Figure 5.40: Four-layered interactive conflict management dashboard design

Figure 5.40 gives a conceptual overview of the conflict management dashboard. At a first glance, it seems very similar to the fashion we visualize metamodel and model differences. However, the details vary considerably. Although the general idea is similar to the interactive visualization shown in Section 5.3.1, the conflict management dashboard is different in the way users are able to interact with it.

In [SMR12], we distinguish between different kinds of interactions. While the difference visualization realizes visual interactions, the conflict management dashboard also features visual interactions that are propagated to the interaction model which finally manipulates the underlying model. Consequently, not the visualization is not only interactive in terms of visual manipulations, but also allows that visual changes are propagated to the respective model. This way it can be considered as a visual domain specific language (VDSL) for model conflicts.

In his bachelor's thesis, Schrade [Sc13] presents tool support that allows to collaboratively resolve conflicts. Thereby, he integrates existing collaboration facilities known from enterprise 2.0, social network, and collaboration platforms to facilitate communication. Kirschner [Ki14] refined this prototypical implementation based on practitioner feedback in his master's thesis. In the present thesis, we describe the final design of the prototype.

DEFINITION 5.9: Collaborative real-time conflict resolution

Collaborative real-time conflict resolution describes the collaborative efforts that aim to resolve conflicts. These include proposals for resolutions, comments, discussions, etc. and can be immediately mediated. ■

In the following, we detail the visual concepts as well as interactions on each of these layers.

5.3.2.1 Layer 1: Metamodel Conflicts

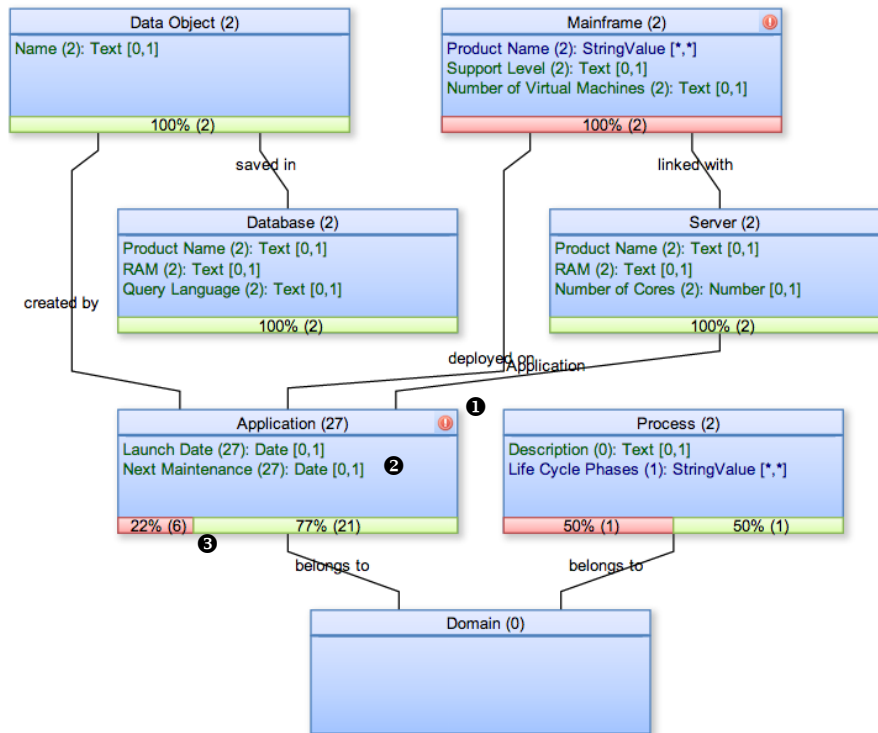


Figure 5.41: Conflict management dashboard: metamodel conflicts

Figure 5.41 shows the first layer of the conflict management dashboard. It shows the conflicts between the different metamodels that have been merged (❶ in Figure 5.41). These conflicts are tasks that refer to specific model elements, i.e. OBJECTDEFINITIONS (❶ in Figure 5.41) and ATTRIBUTEDEFINITIONS (❷ in Figure 5.41). Further, it shows aggregated information about conflicts between different models (❸ in Figure 5.41), i.e. OBJECTS and ATTRIBUTES. The user can either click on an icon denoting an exclamation mark to open a conflict task in order to resolve a metamodel conflict or click on an OBJECTDEFINITION to drill-down on conflicts of the model, i.e. OBJECTS that conform to the selected OBJECTDEFINITION. The latter action opens the next layer of the conflict management dashboard. Thereby, the scope of the viewed conflicts is already narrowed. In our evaluation, EA practitioners regard this way of navigation intuitive (cf. Chapter 7).

5.3.2.2 Layer 2: Model Conflicts

The second layer of the conflict management dashboard is depicted in Figure 5.42. Since the number of elements at this layer is considerably larger than on the first layer, we hide any detail and only indicate that an OBJECT or an ATTRIBUTE thereof is conflicting with another one (❶ in Figure 5.42). The same iconification than on layer 1 applies to layer 2, i.e. if an OBJECT or its ATTRIBUTES are in conflict the user can click the exclamation mark to navigate to the conflict task. Note that the first layer is still visible to the user (❷ in Figure 5.42).

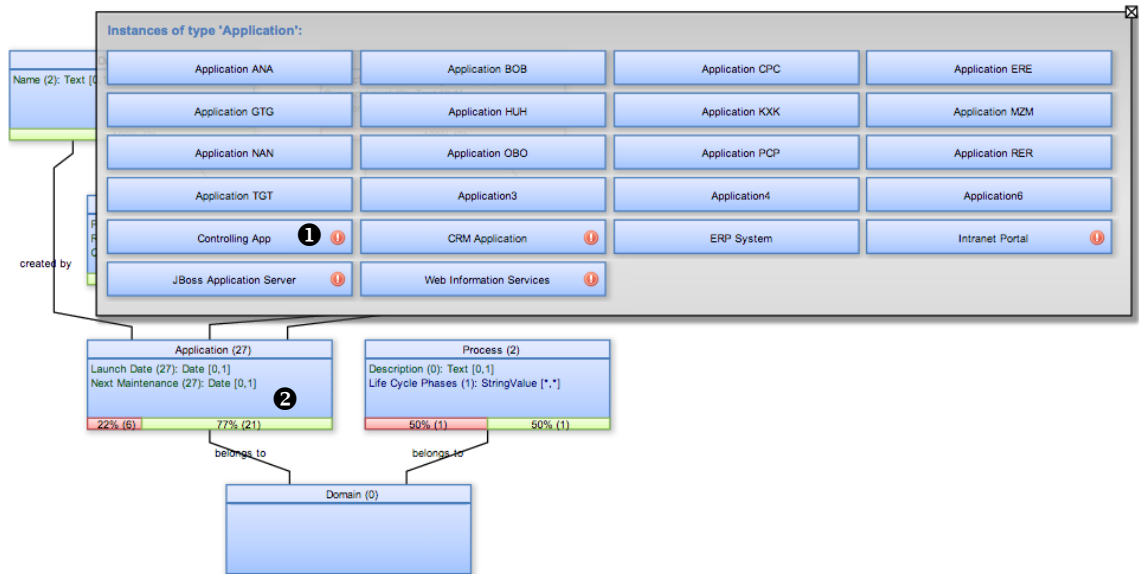


Figure 5.42: Conflict management dashboard: model conflicts

5.3.2.3 Layer 3: Context of Model Conflicts

Figure 5.43 shows the third layer of the conflict management dashboard. It illustrates a conflicting OBJECT (1 in Figure 5.43) and its ATTRIBUTES. This view incorporates further details, i.e. all ATTRIBUTES and those that are conflicting with other changes (2 in Figure 5.43). Additionally, the context of the OBJECT is shown (3 in Figure 5.43). Thereby, the depth of this contextual neighborhood is configurable. The objects are arranged as a hierarchical graph visualization similar to the graph layout applied to layer 1. Note that other layers (4 and 5 in Figure 5.43) are still visible to the user and are accessible at any time. This way, users are able either to roll-up, to drill-down, or to navigate to adjacent model elements.

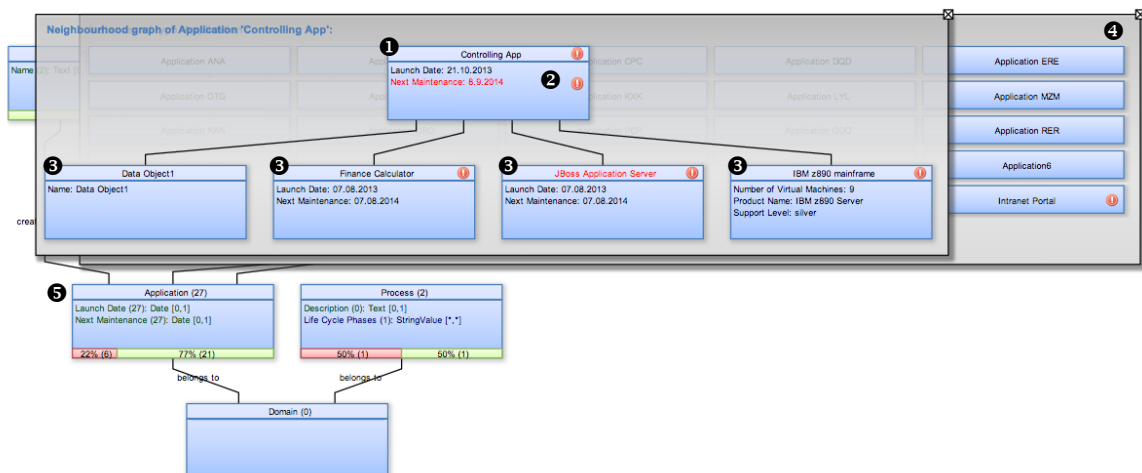


Figure 5.43: Conflict management dashboard: model conflicts in context

5. Federated EA Model Management Design

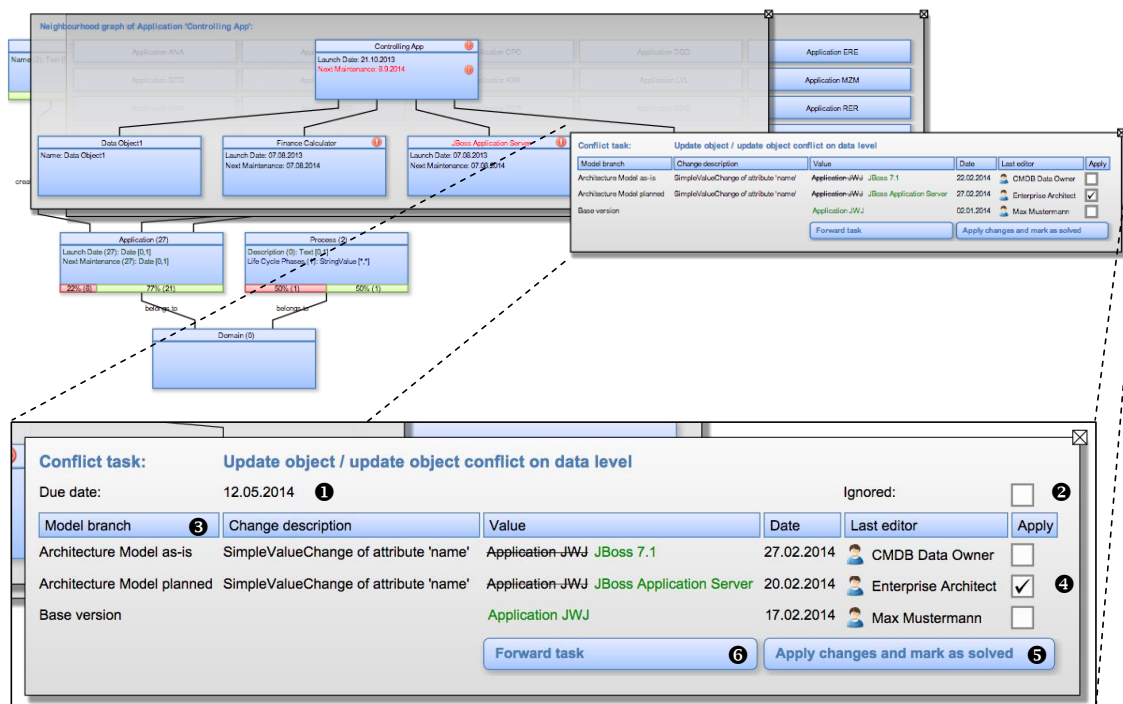


Figure 5.44: Conflict management dashboard: conflicts resolution tasks

5.3.2.4 Layer 4: Conflict Resolution

Figure 5.44 shows the fourth layer of the conflict management dashboard. After drilling down on a specific conflict, this layer allows users to find and apply a resolution. Since every conflict is stored as a task, it also has a due date (1 in Figure 5.44). The task and respective conflict can be immediately ignored by setting a flag (2 in Figure 5.44). All operations applied by different users are visualized in a tabular manner (3 in Figure 5.44). This table lists the source, a description of what was changed, the actual values that were applied, a precise timestamp, and the roles that issued the change.

Since our metamodel allows to store objects, attributes, and values that do not conform to their definitions, i.e. violate constraints, the user can apply any of these values (4 in Figure 5.44). However, in practice, the different involved parties typically agree upon one value that represents the state of affairs in the real world. The changes can be applied to the model with information about the user initially performing the change (5 in Figure 5.44). Each task can be forwarded to another user which enables asynchronous collaboration (6 in Figure 5.44). In the next section, we discuss how the conflict management dashboard allows synchronous collaboration.

5.3.2.5 Collaboration

Figure 5.45 depicts conflict management dashboards of two different users. Via the collaboration button (1 in Figure 5.45), different features are enabled that allow synchronous

collaboration. The conflict management dashboard realizes different collaborative features detailed in the following. First, users must be added to a collaborative session (② in Figure 5.45). The communication can take place via voice (③ in Figure 5.45) or chat (④ in Figure 5.45). Furthermore, different mouse cursors are shown to facilitate the discussion (⑤ in Figure 5.45). This real-time collaboration can be compared with screen sharing. However, different access rights are enforced for each user individually. Interactions on the visualization are synchronized such that each user that has joined the conflict resolution session gets the same state of the visualization with respect to individual access rights. That means, if different access rights apply (⑥ in Figure 5.45), different visualizations are shown. Information that is not visible to all parties is depicted with a different opacity to inform the user about the different access rights on this model element.

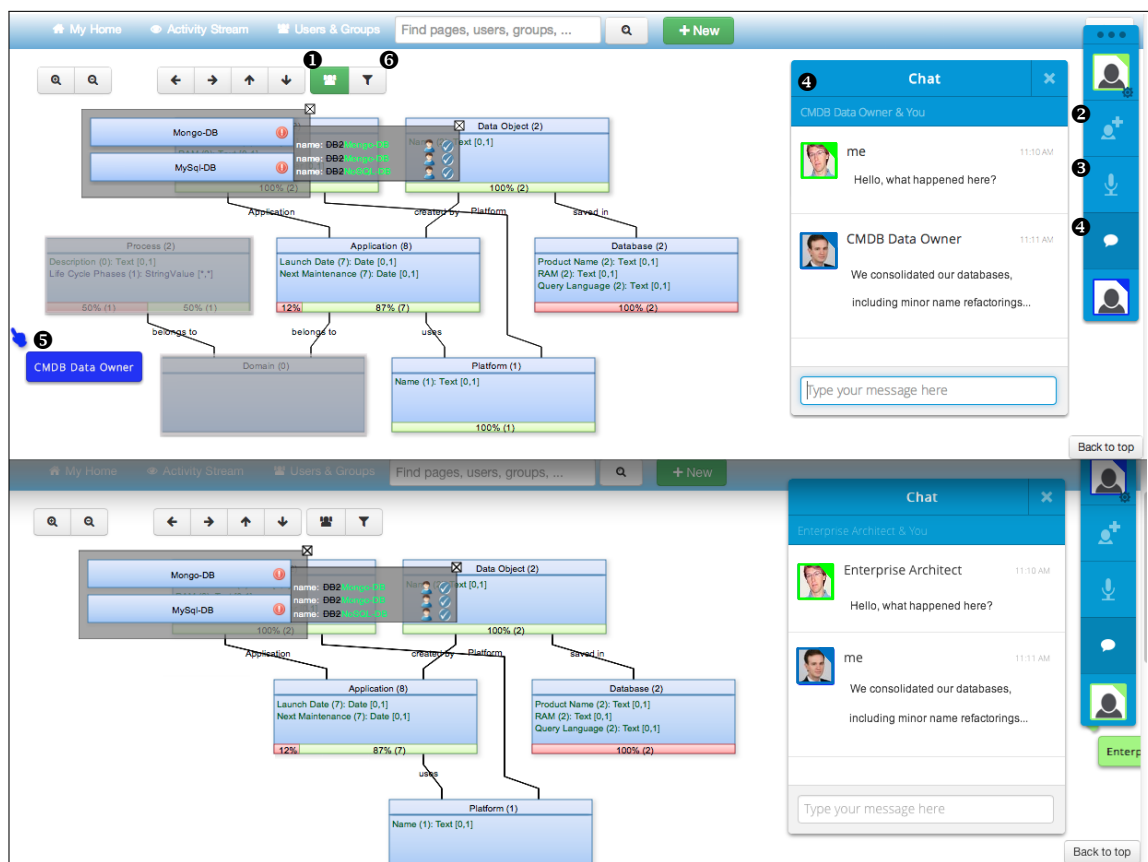


Figure 5.45: Conflict management dashboard: collaboration with distributed access rights

5.3.2.6 Filtering

Figure 5.46 depicts a screenshot of the filter dialog within the prototypical implementation of MODELGLUE. In addition to the conceptual dialog (see Figure 5.41 on p. 208), we added a checkbox to indicate that only model elements for which tasks exist are displayed in the conflict management dashboard (① in Figure 5.46).

For each OBJECT, all attributes are shown (② in Figure 5.46) and depending on the type of this ATTRIBUTE, users can choose from a set of comparators (③ in Figure 5.46) and type in a value (④ in Figure 5.46) that is used to compare the ATTRIBUTE's value with. Filtering rules can be concatenated logically by adding further rules (⑤ in Figure 5.46). Users can either add rules on the same logical level (⑥ in Figure 5.46) or add sub-rules (⑦ in Figure 5.46). Sub-rules are displayed as indented rules (⑧ in Figure 5.46). All rules are logically concatenated with **and** (⑨ in Figure 5.46) or **or** conjunctions (⑩ in Figure 5.46). Users are also allowed to remove rules (⑪ in Figure 5.46).

In a final step, the filter is applied to the conflict management dashboard (⑫ in Figure 5.46). If the conflict management dashboard is in collaboration mode, this filter is propagated to the different parties currently participating in the conflict resolution session. Although the filter is used to query model elements, individual access rights are applied to each view of the conflict management dashboard during the conflict resolution session.

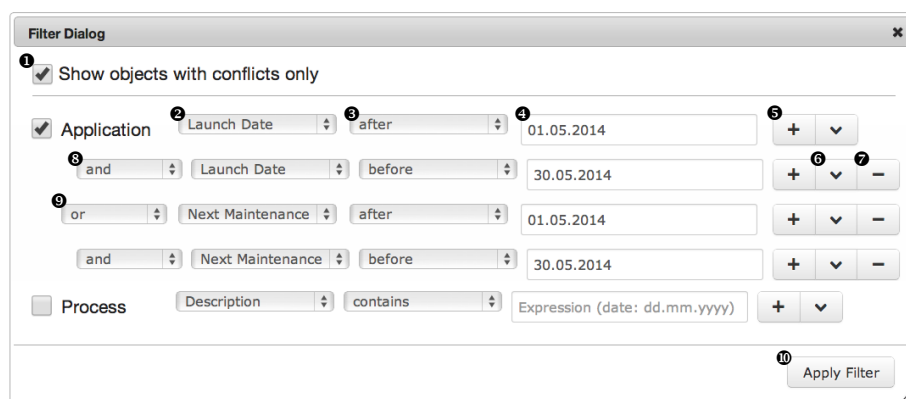


Figure 5.46: Screenshot of the user interface for a dialog to specify filter rules

Our final note on the design of the filter are further design considerations we had during its development. We considered an increase set of interactions for each condition. Our idea was to provide controls for indenting $>$, unindenting $<$, moving up \wedge , or moving down \vee a condition. However, we chose to prefer a lean design over adding additional control elements to the dialog.

5.3.2.7 Printing

MODELGLUE supports face-to-face communication by providing print support for its visualizations (cf. Section 6.3). Figure 5.47 depicts the basic controls of an interactive visualization; as these are straight-forward, we just provide a very brief description. Besides zoom (① in Figure 5.47) and navigation facilities (② in Figure 5.47), the collaboration mode and filter can be toggled (③ in Figure 5.47).

Moreover, the visualization can be downloaded in one of the formats, PPT/PPTX, Open-Document Presentation (ODP), Portable Network Graphics (PNG), or Scalable Vector Graphics (SVG). This way, one can print the visualization and use the visualization in meetings, e.g. to mediate model conflicts. Thereby, it is important that the visual ele-



Figure 5.47: Navigation controls and download menu of the MODELGLUE's visualizations

ments can be manipulated for the non-rasterized formats, i.e. PPT/PPTX, ODP, and SVG (5 in Figure 5.47). This way, one can alter the visualization easily and present ideas or intermediate results that build a starting point for discussions to resolve conflicts.

5.4 Summary

In this chapter we detailed a holistic concept of Federated EA Model Management. Key to its design is an iterative approach to integrate information sources and the notion of tasks to facilitate collaboration. This design is based on the assumption that bi-directional technical interfaces between an EA repository and multiple information sources are too costly. As a solution, tasks are used to resolve conflicts within the federation, i.e. in the EA repository as well as in all integrated information sources. We provide a design to support quality assurance through visual differencing during the iterative Federated EA Model Management process and incorporate tasks in an interactive conflict management dashboard.

We described a process for Federated EA Model Management that is based on results of our research and incorporates feedback from multiple surveys and interview series with industry experts (cf. Chapter 7). This process is used as a vehicle to describe the core principles and design of MODELGLUE—the software-support for Federated EA Model Management. In the next chapter, we give an overview of some important implementation aspects with a focus on the interactive visualizations.

SOFTWARE SUPPORT FOR FEDERATED EA MODEL MANAGEMENT

In this chapter, we reveal implementation details of MODELGLUE. Our implementation is based on the Enterprise 2.0 platform initially developed by Büchner [Bü07] and extended by Neubert [Ne12]. We detail significant parts of the implementation, i.e. extensions of access control, tasks, and interactive visualizations.

6.1 Architecture of MODELGLUE

Figure 6.1 depicts the component architecture of MODELGLUE. Note that for the sake of clarity not all dependencies are displayed and the actual implementation is more complex. In line with Neubert [Ne12, p. 90], we divide the architecture into the classical Model View Controller (MVC)¹ structure to categorize the components by their technical core capabilities.

Model denotes components which are primarily concerned to facilitate modeling and storing information.

View denotes components which are primarily concerned to facilitate the (visual) communication information.

Controller denotes components which are primarily concerned with the flow of events or computations.

From bottom-up, multiple *Federated Systems* which denote the different information sources are shown. We assume a certain structure which is fairly general. Besides the general

¹For a comprehensive description of the MVC pattern we refer to Gamma et al. [GHJ⁺94] and Sommerville [So11, p. 115].

MVC structure, the Federated Systems are divided into general modeling capabilities and specialization which implements domain specific workflows and UIs. For Federated EA Model Management, a central point here is that the model is stored in a persistent manner and accessible via an API. This holds true for the model as well as for the metamodel. An alternative often used in practice is to import (semi-)automated exports that are exchanged through spreadsheets (cf. Chapter 7). In these cases, the metamodel is often hardcoded and communicated on paper or presentations than via an API.

Further, Figure 6.1 depicts a *Federal System* that is presented as a four layered application architecture, i.e. the import, modeling, conflict detection, and conflict resolution layers. The importer connects various information sources and imports model as well as metamodel changes to the import model stored within the Federal System. Depending on the information sources, the differences of a model to its previous version have to be calculated manually if the information sources deliver time slices rather than incremental changes. A differencing widget can be used to view the differences of models which can be either previous versions of a model or different models (cf. Section 5.3.1). The model merger integrates models with each other under the usage of the conflict resolution strategy configured in the conflict resolution strategy widget. During the (collaborative) merge of models, the workflow engine serves to maintain the states of tasks and allows to forward tasks whereas the task manager creates new tasks and determines addressees of tasks employing information stored in the persistency component and the fine-grained access control manager. Tasks can be either accessed via the tabular view denoted conflict list widget or via an interactive conflict management dashboard. This visualization is generated by a visualization generator in multiple model-to-model transformations.

6.1.1 Configuration of Variability Points

Throughout the development, we discussed different variants of the architecture presented above. In the following, we outline the most significant variability points and state how these are configured in the prototypical implementation of MODELGLUE.

Integrated vs. loosely coupled EA repository. The EA repository can be either loosely coupled or integrated directly in the Federal System. In MODELGLUE, we integrated the EA repository and the Federal System into one information system.

Integrated vs. loosely coupled information source. Similarly to the EA repository, an information source can be coupled with the Federal System directly. A directly integrated information source does not need a separate export and import model. In MODELGLUE, we did not integrate any information source directly but rather developed import mechanisms. For each information source, a separate export model outside of MODELGLUE as well as an import model within MODELGLUE exists.

Automated vs. manual exchange of metamodels. In MODELGLUE, we do not exchange metamodels automatically. However, we ‘duplicate’ changes within the information source’s metamodel such that mechanisms for metamodel differencing and conflict detection can be utilized and demonstrated for evaluation purposes.

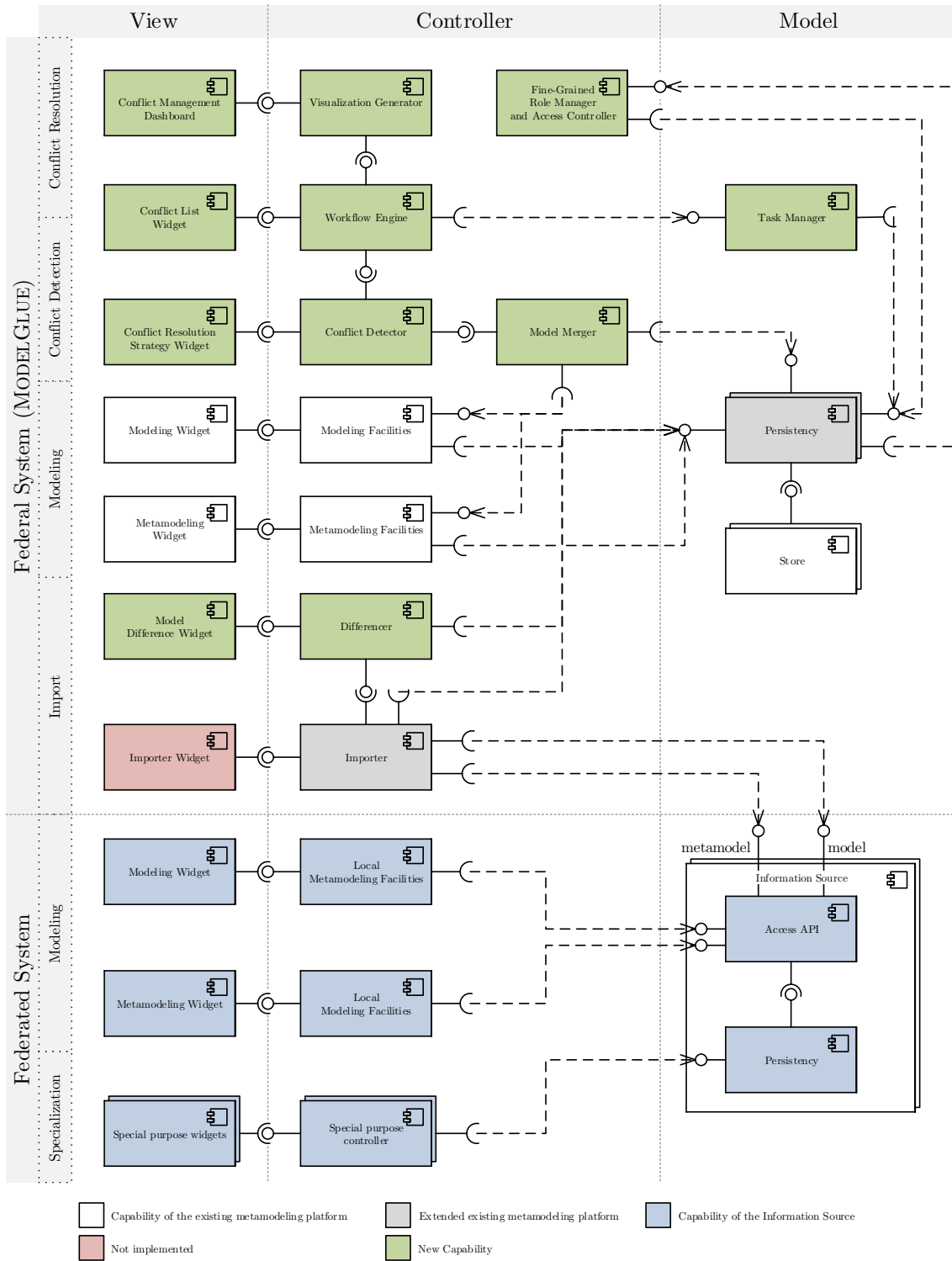


Figure 6.1: Architectural overview of MODELGLUE as UML component diagram

6.2 Extending a Non-rigid Typed Repository with Tasks

In this section, we describe how our conceptual design of MODELGLUE maps to an existing system and present extensions thereto which fit the needs of MODELGLUE.

For the prototypical implementation, we extended Tricia, an enterprise 2.0 platform introduced by Büchner [Bü07] and extended with flexible, i.e. non-rigid typed model capabilities, by Neubert [Ne12]. Neubert explains how the concept of non-rigidity support the bottom-up evolution of models and introduces five stages that guide users from non-rigid to rigid typed model [Ne12, pp. 37,41–43]. His model offers the flexibility that can be exploit to realize a repository that offers the necessary degree of freedom during conflict resolution. Building on Neuberts work, Kirschner [Ki14, p. 21] illustrates a mapping of the conceptual foundations of MODELGLUE to the metamodel of Neubert [Ne12, pp. 18,92]. Thereby, Kirschner illustrates the extensions and concepts reused throughout the prototypical implementation of MODELGLUE. We refer to Kirschner [Ki14, p. 21ff] for an extended version of Neubert’s metamodel (cf. Appendix B).

In the subsequent sections, we detail how the roles and responsibilities are realized in the prototype and how tasks have been made accessible for end users.

6.2.1 Roles and Model Actions

Roles are realized as so called PRINCIPALS within Tricia. The implementation details are extensively discussed by Neubert in [Ne12, pp. 20–22]. He advocates not to define access rights on ATTRIBUTES and ATTRIBUTEDEFINITIONS. To some extent, Neubert’s implementation derives default access rights from a MODEL. For MODELGLUE, however, a more fine-grained access control is necessary. In order to realize the chain of responsibility as proposed in the design of MODELGLUE (cf. Section 5.1.2.4), we introduce a new subclass of PRINCIPAL and incorporated it into the existing system. For the exact implementation details, we refer the interested reader to Kirschner [KR14, p. 21ff]. The changes implemented by Kirschner have several implications on the UI. In the following, we give an overview how one can set access rights in the prototypical implementation of MODELGLUE.

In Tricia, a MODEL is also called *space* or workspace. Figure 6.2 depicts such a workspace and its actions. For each workspace, a principal is held responsible (❶ in Figure 6.2). Further, we extended the actions for a workspace. New operations ❷–❺ in Figure 6.2 are specific to support the Federated EA Model Management process (see Figure 5.6 on p. 143). Branching a model (❷ in Figure 6.2) creates a clone with the reference to its origin. Merge (❸ in Figure 6.2) triggers a manual merge that can be executed according to the merge algorithm as specified in Section 5.2.5. Differences of models can be calculated and visualized (❹ in Figure 6.2) as described in Section 5.2.4. Importing information from an information source stored in an XLS/XLSX spreadsheet that is uploaded to the system can also be triggered manually (❺ in Figure 6.2). Further capabilities that are used to facilitate MODELGLUE with the technical system Tricia, e.g. altering the metamodel, are described by Neubert in [Ne12, ch. 4].

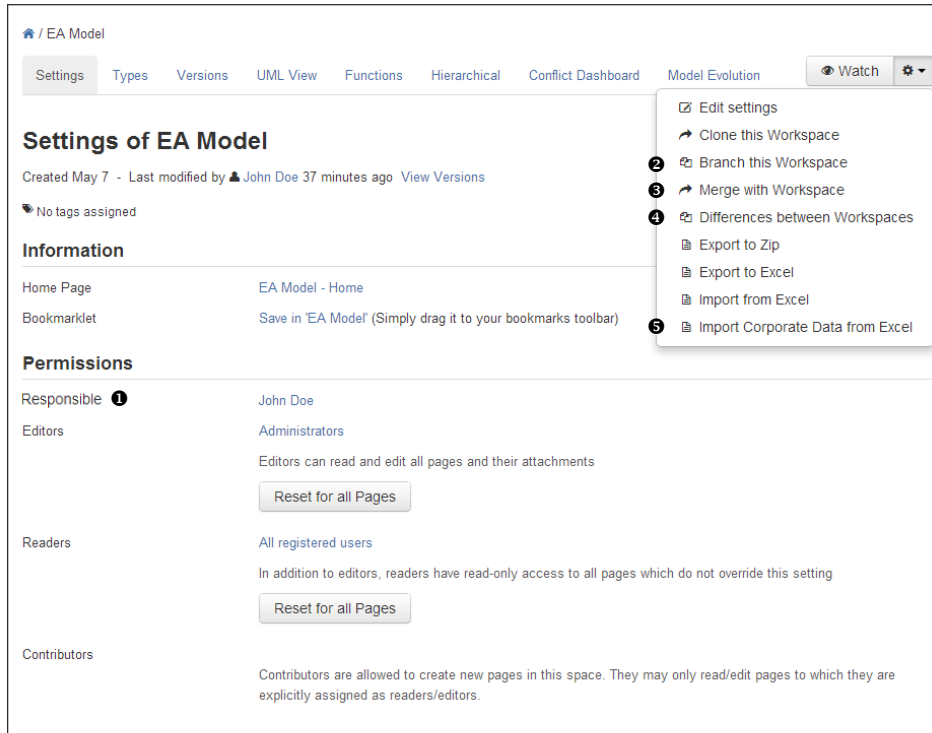


Figure 6.2: Access right dialog of a MODEL

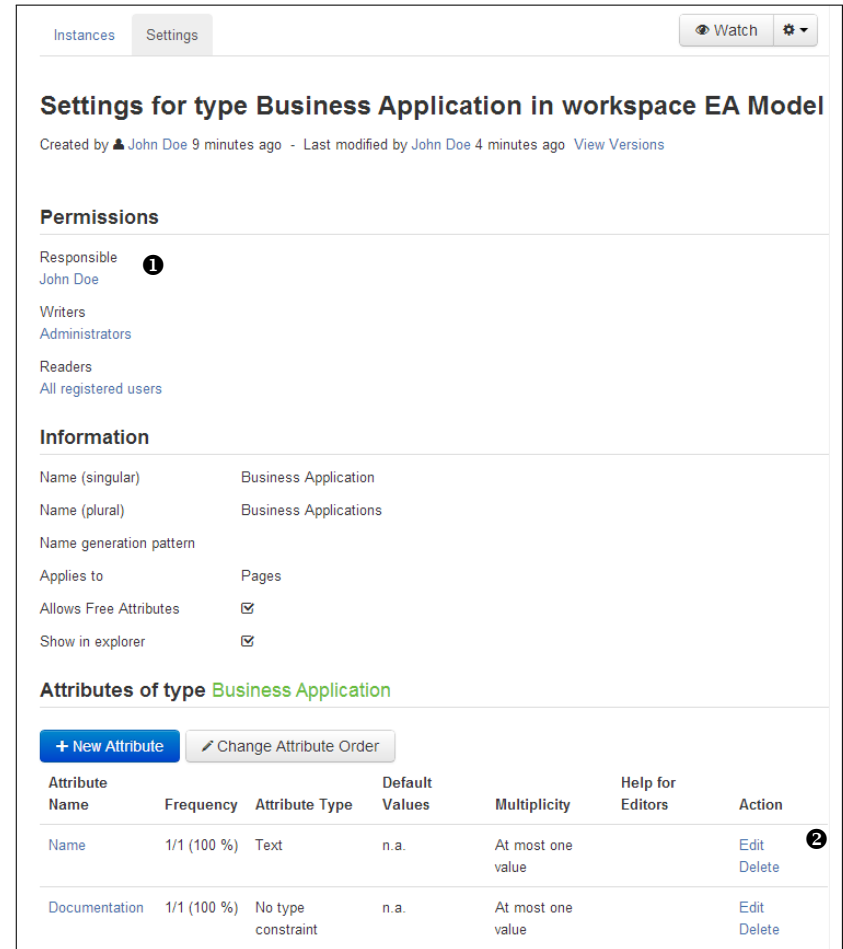


Figure 6.3: Access right dialog of an OBJECTDEFINITION

Figure 6.3 depicts the UI that can be used for managing access rights of a single OBJECTDEFINITION. The permissions (❶ in Figure 6.3) have been altered such that they are applicable for an OBJECTDEFINITION and serve as default access rights for the ATTRIBUTEDEFINITIONS (cf. Section 5.1.2.4). Further, all ATTRIBUTEDEFINITIONS are shown and can be edited (❷ in Figure 6.3). As described in Section 5.1.2.4, the default access rights are inherited from the MODEL, i.e. space in Tricia.

Figure 6.4: Access right dialog of an ATTRIBUTEDEFINITION

Figure 6.4 illustrates the edit dialog for a single ATTRIBUTEDEFINITION. The dialog enables one to specify a responsible role for each attribute (❶ in Figure 6.4). Moreover, it allows to override access rights of the respective OBJECTDEFINITION (❷ in Figure 6.4). Here, the ATTRIBUTE ‘Name’ can only be written by the user ‘John Doe’.

Figure 6.5 shows the access rights dialog for a single OBJECT. The default access rights for an OBJECT are determined by the access rights of its OBJECTDEFINITION.

Figure 6.6 shows the edit dialog and the access rights for a single OBJECT. As illustrated, only on a click on the ‘Permissions’ button, the access rights are shown. In most cases, the access rights on ATTRIBUTE and ATTRIBUTEDEFINITION stay the same. However, practitioners find this configuration of access rights for one ATTRIBUTE meaningful. From a user perspective, the access rights are copied from the ATTRIBUTEDEFINITION. If these do not exist, the access rights of an OBJECT or OBJECTDEFINITION are used. The user can then override these defaults with custom access rights for a specific ATTRIBUTEDEFINITION. Internally, the system implicitly detects if default access rights are restored by the user. From a user perspective, only effective access rights are shown such that the complexity to find these effective access rights and where they originate from is hidden entirely.

Figure 6.7 shows the dialog to invoke the model differencing. This includes the calculation of differences and subsequent visual analysis. The current model is already configured as

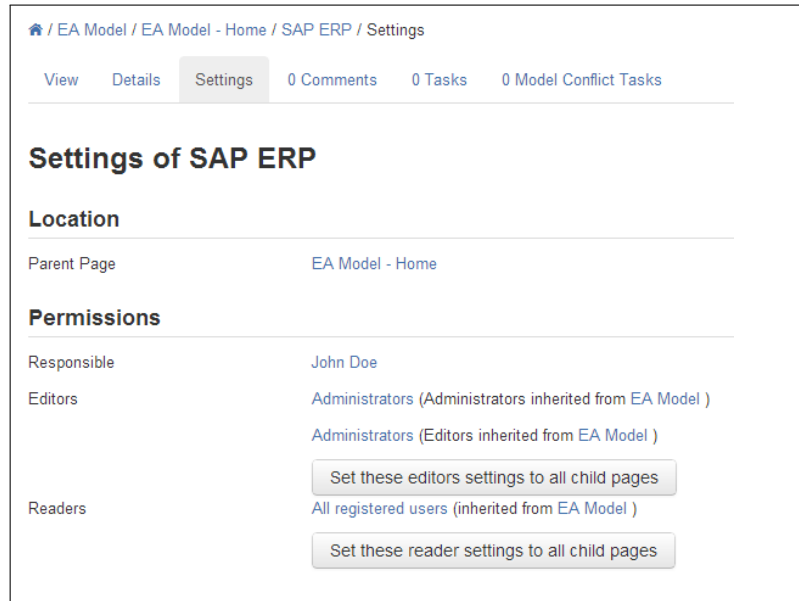


Figure 6.5: Access right dialog of an OBJECT

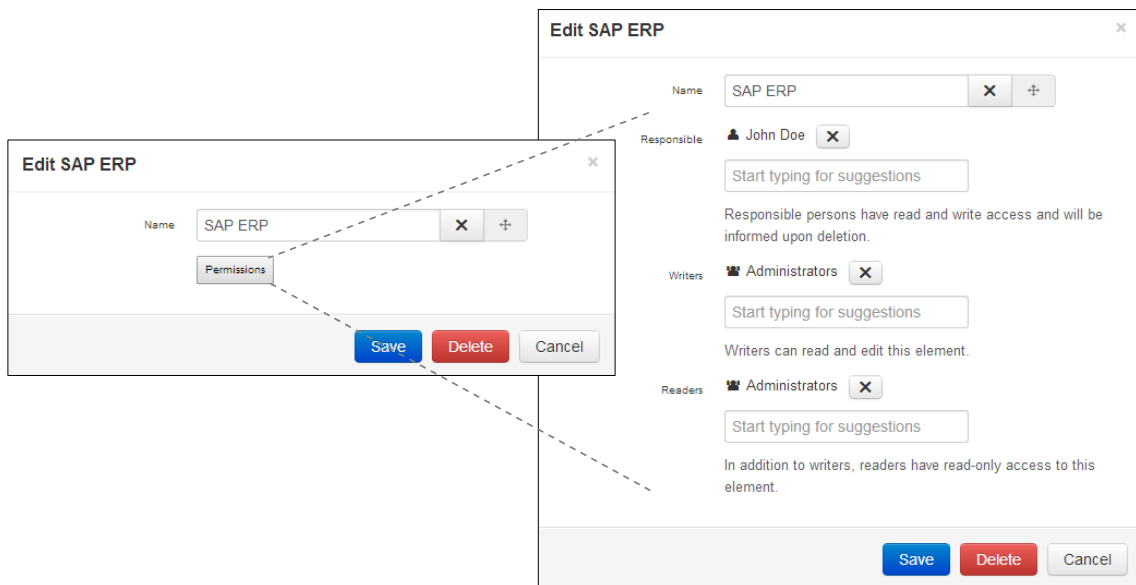


Figure 6.6: Access right dialog of an ATTRIBUTE

default for one side (❶ in Figure 6.7), whereas another model must be entered by the user (❷ in Figure 6.7). Thereby, the autocomplete feature of Tricia (cf. [Ne12, p. 112]) supports the user such that only spaces, i.e. models, are proposed.

In Figure 6.8, we depict the UI to configure the merge of different MODELS. By default, the current workspace is already selected as one of the source models and as the target of the merge action (❶ and ❸ in Figure 6.8). The user is supported by the autocomplete feature of the system during the input of the models that are merged (❷ in Figure 6.8).

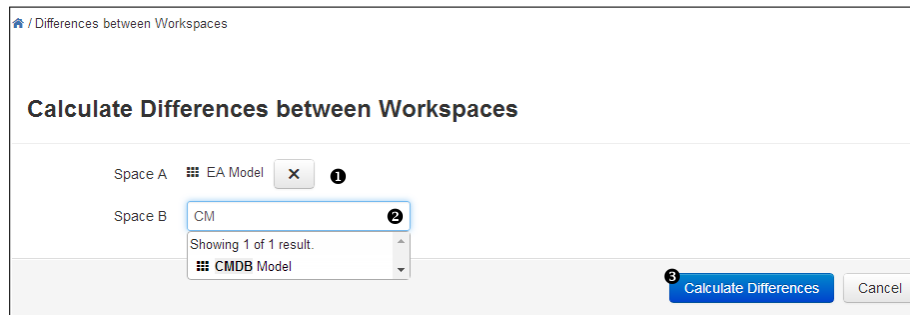


Figure 6.7: A dialog to configure the differencing of MODELS

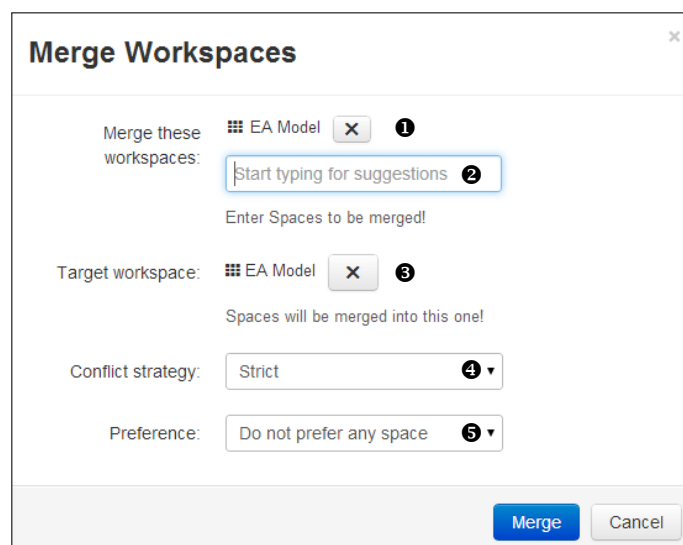


Figure 6.8: A dialog to configure a merge of MODELS

Moreover, users can choose a conflict resolution strategy that is applied throughout the merge (4 in Figure 6.8). Besides this strategy, one might prefer a model over another during conflict resolution. This preferred model can be specified in 5 in Figure 6.8. The dialog does invoke a long-lasting merge process. For this process, the target model is transferred to a preview workspace that is used during the conflict resolution. If anything goes wrong, e.g. the enterprise architect configured a customized merge rule (cf. Section 5.2.8.4) that deletes ‘random’ model elements, the preview workspace can be deleted and no effective changes to the involved models are made. Further details on the UI design can be found in [Ki14, ch. 3].

6.2.2 Tasks

In MODELGLUE, we rely on tasks to 1) enable collaboration among different parties, 2) store information on a conflict, 3) realize a flexible conflict resolution process. Besides the interactive conflict management dashboard, we provide a tabular view on model conflict tasks. Figure 6.9 shows this view of model conflicts within Tricia. In our initial design,

Date and time of merge: Mar 12 **1**

Conducted by: Enterprise Architect

Spaces involved: Architecture Model as-is, Architecture Model planned

Target space: Architecture Model planned

Tasks assigned to you: **2**

Page	Status	Related To	Classification	Attribute	Branch A: Architecture Model as-is	Branch B: Architecture Model planned	Base Version	Action
ConflictTask	<input type="checkbox"/>	Web Information Services	UPDATE OBJECT / UPDATE OBJECT Conflict on DATA level	name	Internet	Web Information Services	Application FSF	<input type="checkbox"/> 6
ConflictTask	<input type="checkbox"/>	CRM Application	UPDATE VALUE / UPDATE VALUE Conflict on DATA level	Next Maintenance	11.02.2015	18.03.2015	11.11.2014	<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
ConflictTask	<input type="checkbox"/>	JBoss Application Server	UPDATE OBJECT / UPDATE OBJECT Conflict on DATA level	name	JBoss 7.1	JBoss Application Server	Application JWJ	<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>
ConflictTask	<input type="checkbox"/>	Controlling App	UPDATE VALUE / UPDATE VALUE Conflict on DATA level	Next Maintenance	20.11.2014	08.10.2014	21.10.2014	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
ConflictTask	<input type="checkbox"/>	Intranet Portal	UPDATE OBJECT / UPDATE OBJECT Conflict on DATA level	name	Intranet Portal	Intranet	Application IVI	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
ValidateTask	<input type="checkbox"/>	Finance Calculator	UPDATE OBJECT / MOVE OBJECT Conflict on DATA level	name	Deprecated Applications	Finance Calculator	Application SFS	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> 7

6 rows total

Show 10 rows | 1 of 1

Tasks assigned to you as member of the Administrators group: **3**

Page	Status	Related To	Classification	Attribute	Branch A: Architecture Model as-is	Branch B: Architecture Model planned	Base Version	Action
ApproveTask	<input type="checkbox"/>	IBM z890 mainframe	UPDATE OBJECT / DELETE OBJECT Conflict on DATA level	name		IBM z890 mainframe	IBM z890 (deletedId=zcnhungp9xag)	<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
ApproveTask	<input type="checkbox"/>	IBM z800 mainframe	DELETE OBJECT / UPDATE OBJECT Conflict on DATA level	name		IBM z800 mainframe		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
ConflictTask	<input type="checkbox"/>	Product Life Cycle Management	UPDATE VALUE / UPDATE VALUE Conflict on DATA level	Life Cycle Phases	Conception Design Realization Service	Conceive Plan Design Realize Service	Conceive Design Realize Service	<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>

3 rows total

Show 10 rows | 1 of 1

Figure 6.9: Tabular overview of model tasks within Tricia

6. Software Support for Federated EA Model Management

we regard this representation not particular suitable to facilitate the resolution of conflict since the context information is not represented visually. However, during the evaluation, one practitioner specifically referred to this tabular overview as an intuitive way to resolve conflicts. The task overview includes meta-information of the merge action (❶ in Figure 6.9). Further we show tasks that are directly assigned to a person (❷ in Figure 6.9) and through membership in a group (❸ in Figure 6.9). For each task, the different involved branches of an EA model are shown (❹ in Figure 6.9). Within Federated EA Model Management, an import model is a branch of the EA model that is synchronized with an information source (cf. Section 5.2.2). The user can choose from different changes of a task (❺ in Figure 6.9) and subsequently the task can be marked as resolved (❻ in Figure 6.9). An approve tasks allows to revert changes if necessary (❼ in Figure 6.9). Note that information on this level is only presented partially as the UI hides some details.

Due Date	May 15	❶						
Creator	Enterprise Architect							
Assigned To	Administrators							
Status	<input type="checkbox"/>							
Related To	Product Life Cycle Management	❷						
Send Email Notifications	<input type="checkbox"/>	❸						
Description		Edit						
Conflict Classification	UPDATE VALUE / UPDATE VALUE Conflict on DATA level	❹						
Conflicting Elements	Product Life Cycle Management in space Architecture Model as-is Product Life Cycle Management in space Architecture Model planned	❺						
Conflicting versions:								
Object name	Object type	Space	Time	User	Change name	New value	Previous value	Action
Product Life Cycle Management	Page	Base version	27.02.2014 10:58	Max Mustermann	Base version	Conceive Design Realize Service		❻ Apply this version!
Product Life Cycle Management	Page	Architecture Model as-is	18.03.2014 12:10	CMDB Data Owner	HybridPropertyChange of attribute 'Life Cycle Phases'	Conception Design Realization Service	Conceive Design Realize Service	Apply this version!
Product Life Cycle Management	Page	Architecture Model planned	01.03.2014 09:42	Enterprise Architect	HybridPropertyChange of attribute 'Life Cycle Phases'	Conceive Plan Design Realize Service	Conceive Design Realize Service	Apply this version!

Figure 6.10: Tabular view of conflict task details within Tricia

Another dialog shows these details of a task. In Figure 6.10, we illustrate this dialog. Similar to the tabular overview dialog, it shows relevant meta-information for a task (❶ in Figure 6.10). Further the model element affected by the changes is given (❷ in Figure 6.10). During the evaluation, one practitioner noted that notifications via e-mail could be incorporated since their collaboration culture heavily relies on email. This feedback has been incorporated in the final design of the tasks (❸ in Figure 6.10). Further, a classification of the conflict that has been detected is shown (❹ in Figure 6.10) as well as a direct link to the involved

elements (⑤ in Figure 6.10), which allows to view the actual information of that element in context. The different changes with their details, e.g. the user that issued the change, the date and time, as well as the implied modifications of the change, are shown. Similar to the tabular overview of model conflicts, a user can apply changes (⑥ in Figure 6.10).

6.3 A Framework for Interactive Visualizations

In [SMR12] we detail a conceptual framework that builds the foundation for the implementation we present subsequently. The conceptual framework builds on the works of Wittenburg [Wi07, p. 89ff and p. 131ff]. Although Wittenburg presents a comprehensive Visualization Model [Wi07, p. 115ff], his model does not include interactive elements.

Special to the EA management discipline is the evolutionary characteristic of an EA model and its metamodel (cf. Section 2.1.4). Commonly, at the beginning of an EA endeavor, questions (i.e. concerns) are not entirely clarified or not of interest, yet [RHM13a]. Thus, the underlying framework is developed in a loosely coupled manner to be able to visualize information that conforms to arbitrary metamodels.

The presented concepts in Section 5.3 employ mechanisms to drill-down information by visual interactions. In information visualization, this is sometimes also called *semantic zoom*.

DEFINITION 6.1: Semantic zoom[†]

Semantic zoom describes the ability of an interactive visualization to add (contextual) information when zooming in or respectively remove (contextual) information when zooming out.

[†]Synonym(s): adaptive zooming [CG02] ■

In the remainder of this section, we detail how semantic zoom is realized within our prototypical implementation. Thereby, the combination of visual elements with interactions poses a particular software engineering challenge for the tool support of Federated EA Model Management. First, we present the architecture of the visualization component and outline general principles. Then, we detail how the visualization framework can be applied to implement an interactive conflict management dashboard introduced in Section 5.3.2.

6.3.1 Architecture

Figure 6.11 sketches the architecture of the visualization framework implemented to realize the visualization component of MODELGLUE. In the following we describe core concepts therein.

- A `VISUALIZATIONOBJECT` denotes every object that is included in a visualization. It carries an identifier that is unique within the visualization component. This way, all visual or interaction concepts can be traced to `VISUALIZATIONOBJECTS`.

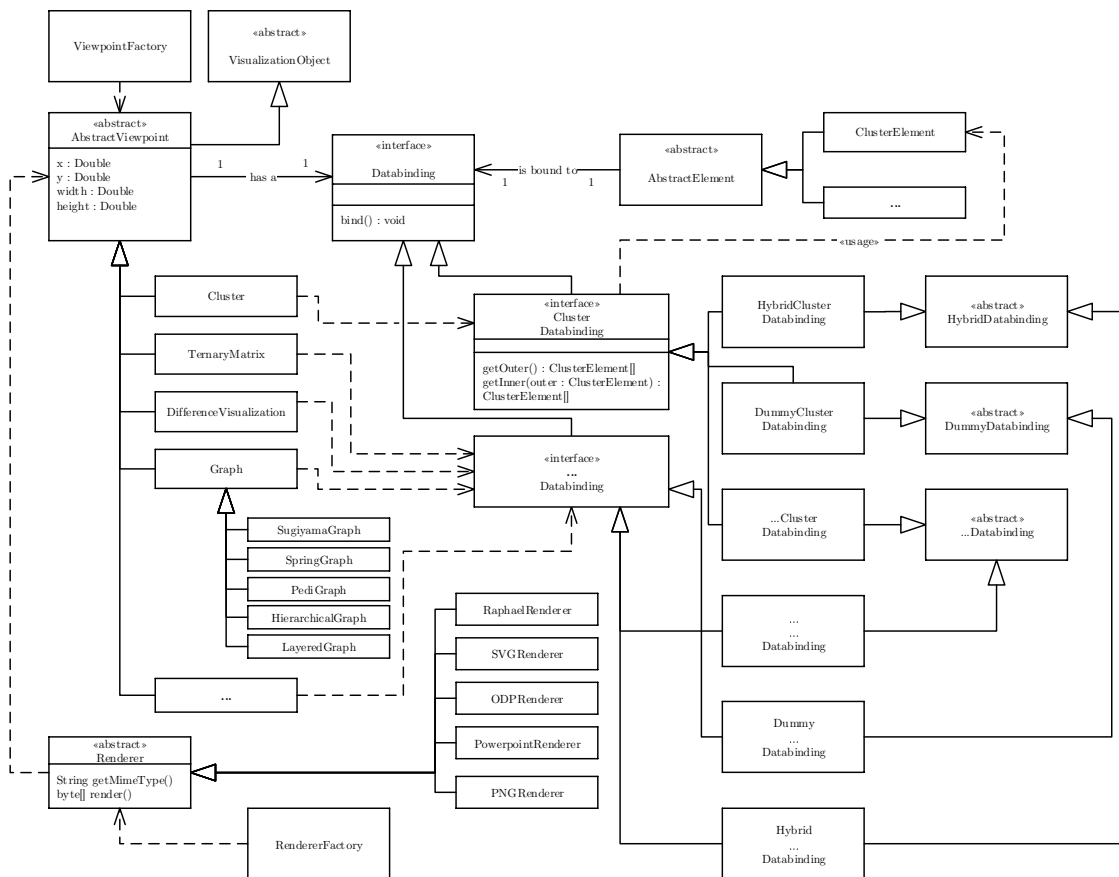


Figure 6.11: Architecture of the Visualization Component of MODELGLUE

- A **VIEWPOINTFACTORY** which serves as a registry for viewpoints that are available. These **ABSTRACTVIEWPOINTS** can be fetched by name. The factory does not return instances but only the class of the respective **ABSTRACTVIEWPOINT**.
- A **RENDERERFACTORY** serves as a central registry for **RENDERERS**.
- The abstract class **RENDERER** performs operations of the rendering process which are common for all target formats. Its subclasses perform any format specific actions including preprocessing and postprocessing of a visualization.
- An **ABSTRACTVIEWPOINT** is a special **VISUALIZATIONOBJECT** that denotes a visualization type (cf. [RZM14, p. 35]) that can be instantiated on its own, i.e. its subclasses implement concrete viewpoints which can be instantiated to views (cf. Section 2.1.2).
- An **ABSTRACTELEMENT** is a unique object identified via a URI and carries a name that serves as a label for visual objects that represent this element. The general idea is that subclasses of the **ABSTRACTELEMENT** implement visualization specific structures that are convenient to traverse in the specific visualization algorithm.

- The interface `DATABINDING` denotes subclasses that transform information represented in a source format to a target structure that is convenient to traverse in the viewpoint. Within the `bind()` method of each subclass, this model-to-model transformation is performed. The binding performs queries on the source model, applies filters (cf. Section 5.3.1.5) and transforms the model into a *view model*. This view model denotes the intended part of a model that is visualized later on. The `DATABINDING` subclasses are aware of any filters that can be applied on both, the source and target structure of the model-to-model transformation.

For each visualization type, a concrete `DATABINDING` exists. This concrete `DATABINDING` may exist for different formats. Thus, each concrete `DATABINDING` implements its interface and extends an abstract class, e.g. `HYBRIDDATABINDING`, which can be employed to send queries to a concrete format. Thereby, the `DUMMYDATABINDING` denotes a format that generates Plain Old Java Object (POJO) model elements which serve as a stub (or dummy) for development purposes. This way, development is considerable faster since developers do not have to provide concrete information and they can abstract from the underlying information store. A concrete example is given by the `CLUSTER`. It accesses a concrete `DATABINDING`, e.g. the `HYBRIDDATABINDING`, through the interface `CLUSTERDATABINDING` and uses the methods defined therein to traverse information.

After describing the core concepts of the visualization framework, we introduce some additional classes that make up the processing pipe.

6.3.2 A Model for Interactive Visualizations

The abstract *visualization model* incorporates the notion of interactivity and visual symbols. Wittenburg [Wi07, p. 119] also presents a visualization model in his PhD thesis. He focuses on mere drawing aspects and the model driven generation of visualizations and, thus, his approach does not incorporate interactivity. In the following section, we refine his model and subsequently detail how interactions are added to this extended visualization model.

Besides the framework of Wittenburg, many frameworks exist that can be employed to foster user interaction. For instance:

- fat client solutions that run as standalone application
 - cross-platform user interface components, e.g. Java Swing [LEW⁺02].
 - interface components that incorporate visualization capabilities, e.g. Java Abstract Window Toolkit (AWT) [Zu97],
 - frameworks, tools, and facilities for the development of an arbitrary visual domain specific language (DSL), e.g. EMF [SBP⁺09],
 - ...

- web based solutions that allow users to view the interface of the application within a browser
 - server-based processing and/or programming followed by web-based rendering of complex UI elements, e.g. JavaServer Faces (JSF) [Be04] or Google Web Toolkit (GWT) [HT07],
 - server-based or script-based approaches to generate an object graph and render visualizations on the client-side, e.g. [RHZ⁺13] or Raphaël [Ba13],
 - ...

While current frameworks separate visual and interaction aspects rather strictly, the abstract visualization model presented below uses a common model for both aspects. Moreover, current frameworks to either concentrate on mere drawing aspects or focus on the implementation of VDSLs. In line with the latter category, we opt to view at a visualization of an EA as a model-to-model transformation. More precisely, as multiple model-to-model transformations. In [SMR12] we presented the interrelationships of the *abstract view model*, view model, and the *view data model* with the *abstract visualization model*, the visualization model, and the *symbolic model*. In this publication we concentrated on giving an overview of the interplay of different models in the course of model-based, interactive visualizations for EA management. In the present thesis, we present details of the visualization model in Figure 6.12 and assume the reader is familiar with the general idea to generate an EA visualization in a model-driven manner presented in [SMR12].

Although incorporated entirely, for a better understanding, we separate the explanation of the abstract visualization model. In a first step, we explain the mere drawing aspect of the visualization model. Before we proceed, we introduce an important term in this context. In computer vision and information visualization, a *scene graph* is an “ordered collection of grouping nodes and other nodes. Grouping nodes [...] may have children nodes.” [In97]. In our framework, a scene graph is represented as an *object graph*.

DEFINITION 6.2: Object graph

An object graph is a directed graph of objects that may include cycles. Objects thereby are virtual items represented within memory and may or may not correspond to real-world objects. ■

- The `VISUALIZATIONOBJECT` is a uniquely identifiable object within an instantiated visualization. It is central to the visualization model and can present 1) an entire visualization, 2) visual elements or composites of visual elements, 3) interactions or composites thereof. Through an adapter, it can pose as either a `COMPOSITESYMBOL` or a `SYMBOL`. This way, the dimensions of an interactive visualization can still be manipulated while holding interaction elements. The interplay with interactions is further detailed below.
- The `COMPOSITEVISUALIZATIONOBJECT` serves as a container realizing the composite pattern (see Gamma et al. in [GHJ⁺94, p. 163ff]). It serves to view interactions and visual elements as one container and realized the object graph in `MODELGLUE`.

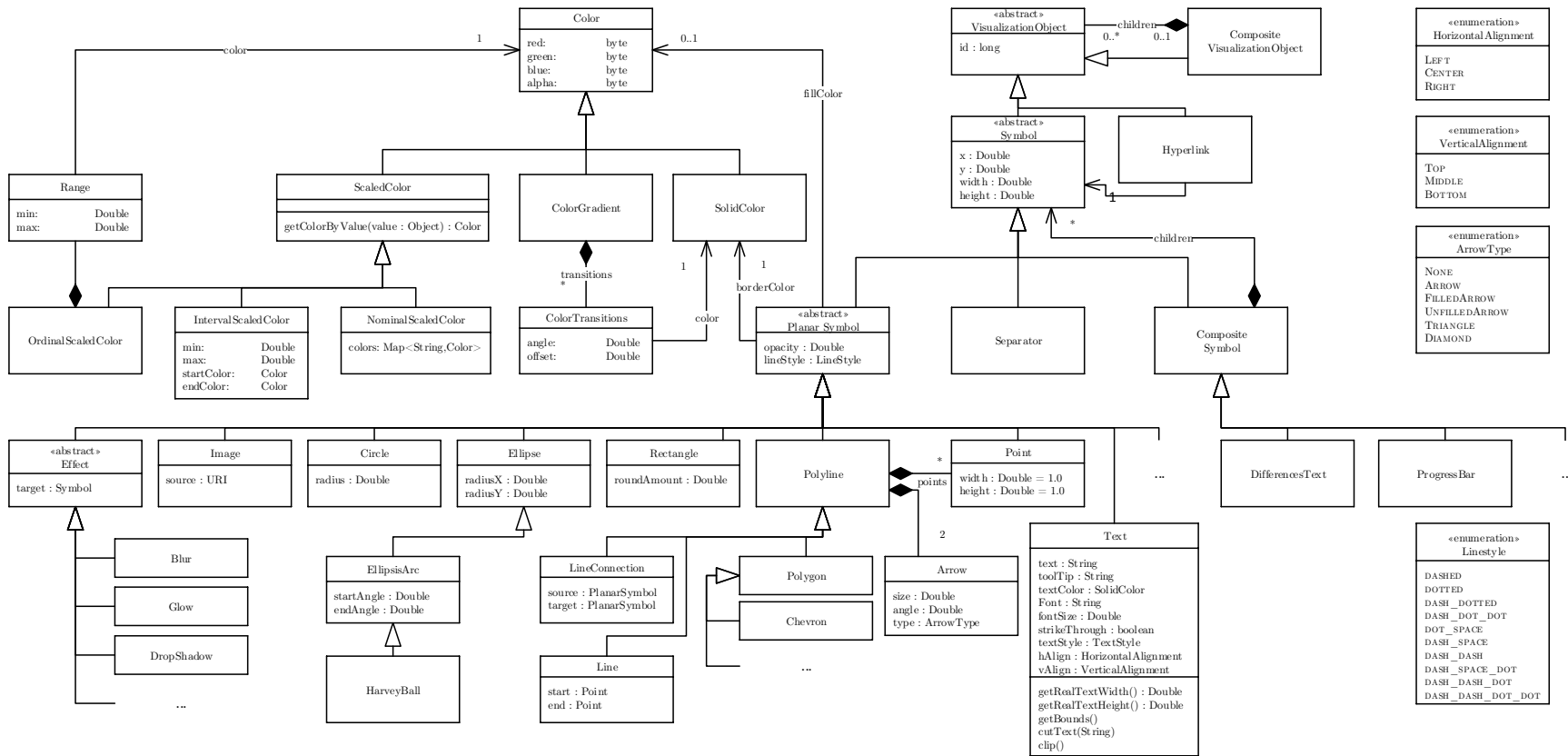


Figure 6.12: Abstract visualization model for interactive visualizations

- `SYMBOL` is similar to the `VISUALIZATIONOBJECT`, however, limited to visual elements that do not carry any interaction. `RENDERERS` that do not support interactions and are rather meant for printing purposes, e.g. `PNGRENDERER` or `POWERPOINTRENDERER`, process `SYMBOLS` only. A `SYMBOL` further can feature a `HYPERLINK`. We regard `HYPERLINKS` not necessarily as an interaction. The interacting nature of `HYPERLINKS` is format specific, e.g. in common browsers, a click on an object with a link means following that link to its destination. However, since we also support formats that do not support interactions, `HYPERLINKS` can also be represented through mere visual concepts, e.g. footnotes within a text.
- `COMPOSITESYMBOLS` implements the composition pattern with the class `SYMBOL` as its elements. It implements a variety of utility functions to move a set of visual elements through the two dimensional space. This particular function is implemented in two different ways. One manipulates an object by a relative position whereas the other adds or subtracts an absolute measure to all visual elements.
- `PLANARSYMBOLS` are two dimensional visual symbols that users can actually have a look at. Its position is given as a tuple of $(x|y)$ coordinates that describe the top left corner of a visual object. Additionally, each `PLANARSYMBOL` has dimension it spans in the two dimensional space determined by its width and height. The class offers additional functions to translate, rotate, and scale visual elements. Other parameters of the `PLANARSYMBOL` are concerned to determine the appearance of the border (color and width) of a visual object, its opacity, fill color, etc.
- `SEPARATOR` is a `SYMBOL`, because it is visible visually but does not feature properties common to a `PLANARSYMBOL`, i.e. it represents for instance empty space within a table cell.

In contrast to Wittenburg [Wi07, p. 135], visual elements, i.e. instances of `SYMBOL`, comply to a coordinate system with the origin $P(0|0)$ at the top left as commonly for other state-of-the-art visualization frameworks [Ba13, BOH11, BH09, HA08, HCL05].

Figure 6.12 also illustrates the different color types. Starting with the abstract class `COLOR`, we represent a single color in class `SOLIDCOLOR`, realize gradients within the class `COLORGRADIENT` that combines several `COLORTRANSITIONS` with different offsets to a composition. Example 6.1 gives an impression how the gradient is calculated based on the different `COLORTRANSITIONS`.



EXAMPLE 6.1: `ColorGradient` with four `ColorTransitions`

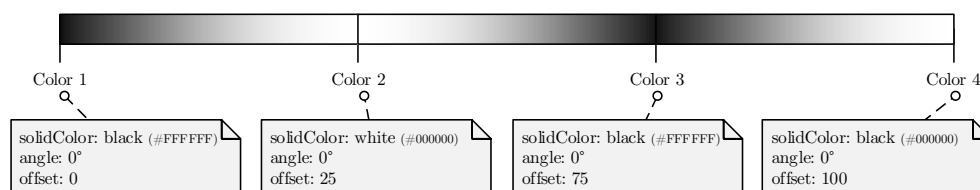


Figure 6.13: `COLORGRADIENT` with four `COLORTRANSITIONS`

Note that the angles in Figure 6.13 are given in degrees whereas the `SOLIDCOLOR` offers a finite set of color constants or uses Hypertext Markup Language (HTML) color codes; the alpha channel of a `SOLIDCOLOR` is set to 100% by default.



Often, the specific color assigned to a visual element is determined by a particular value. Such a behavior is realized within the class `SCALEDCOLOR`. The actual color that is assigned to an object depends on the value passed to the `getColorByValue` method. Subclasses of `SCALEDCOLOR` refine how this calculation is performed. In case of the class `INTERVALSCALEDCOLOR` the calculation of the color depends on a range between `startColor` (assigned to `min`) and `endColor` (assigned to `max`). Intermediate colors for values between `min` and `max` are calculated such that `COLORS` for values in a continuous manner are returned. Another color type is `ORDINALSCALEDCOLOR`. In contrast to the `INTERVALSCALEDCOLOR` type it has several ranges a value could fall into. Depending on the range a value is in, the respective `COLOR` gets assigned. Thus, it does not calculate colors in a continuous manner but returns rather discrete colors. The final color type is `NOMINALSCALEDCOLOR`. While values passed to this class may be numerical the `NOMINALSCALEDCOLOR`, looks up the assigned color for a particular value in a map and, thus, could also be considered a simple LUT for colors.

When drawing different symbols as a group, it is beneficial to get the dimensions $d(x, y, h, w)$ of an entire container. This is realized by the next composite pattern whereas `SYMBOL` is the component, the `COMPOSITESYMBOL` serves as composite, and subclasses of `PLANARSYMBOL` serve as leafs.

Before we explain the subclasses of `PLANARSYMBOL`, we detail how the absolute width S_w^{abs} and height S_h^{abs} of it is calculated in Equation 6.1.

$$\begin{aligned} S_w^{abs} &= S_w + 2 * S_{bw} \\ S_h^{abs} &= S_h + 2 * S_{bw} \end{aligned} \tag{6.1}$$

Whereas S_{bw} denotes the border width of a symbol and S_w and S_h denotes the width and height of a `PLANARSYMBOL`. As illustrated, the border width must be added to get the absolute size of a `PLANARSYMBOL`. In line with Wittenburg [Wi07] and Mykhashchuk [My11], we outline the most important subclasses of `PLANARSYMBOL`, their essential properties, and the most important methods.

`TEXT` is used to represent literals within the visualization. The methods, `getRealTextWidth()` and `getRealTextHeight()` are often used for layouting purposes during the transformation within a visualization. These methods can be used to determine the actual height and width of a text with a given horizontal and vertical alignment, font, etc. If a `TEXT` object requires more space than specified by its width, i.e. it requires more space than it should, it gets shrunk to its width. We exemplify this technique below.

**EXAMPLE 6.2: Cutting text to a specific width**

A TEXT object with width=2cm can only contain several literals in a visual representation. Figure 6.14 shows the original text and the text after it has been cut by the algorithm that is applied within the visualization framework.

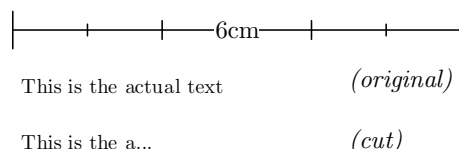


Figure 6.14: Cutting literals within the visualization framework



Note that a tooltip is used to show the user the original text. POINT is a small dot with predetermined width and height values that are fixed. A POINT also carries two coordinates. Thus, it is also used by the POLYLINE to specify the path of a line. POLYLINE is a line that is made up of a path which is not closed. The POLYLINE further comes with different LIFESTYLES and features two ARROWS (star and end) that initially are set to null. POLYGON subclasses may realize more sophisticated shapes such as CHEVRONS. Mykhashchuk [My11] details substructures often seen in information visualization that are POLYGONS. Although she classifies the CHEVRON as a SYMBOL, it can be implemented best as a POLYGON in our framework. In contrast to other PLANARSYMBOLS, the position of a POLYGON and POLYLINE is determined as follows. Let $P(p^x|p^y)_{1..n}$ be POINTS of a POLYGON S , then the position and dimensions of S are calculated as shown by Equation 6.2.

$$\begin{aligned}
 S_x &= \min(p_{1..n}^x) \\
 S_y &= \min(p_{1..n}^y) \\
 S_w &= \max(p_{1..n}^x) - S_x \\
 S_h &= \max(p_{1..n}^y) - S_y
 \end{aligned}
 \tag{6.2}$$

The LINECONNECTION is a straight connection between a SYMBOL s_1 at $(x_1|y_1)$ and another SYMBOL s_2 at $(x_2|y_2)$. The connection between these symbols is calculated in the course of the transformation of the visualization such that it is not required to calculate $\{(x_1|y_1), (x_2|y_2)\}$ during design time of a visualization explicitly which gives developers some freedom when connecting SYMBOLS whose position is not yet determined. LINE in contrast to a LINECONNECTION, needs two coordinates $\{(x_1|y_1), (x_2|y_2)\}$ that must be explicitly stated at design time. RECTANGLE is a rectangular shape specified by its position at the upper left corner with a $(x|y)$ tuple, and its width and height. CIRCLE is a round shape with a radius. ELLIPSIS is a round shape with two different radius values. One can specify the radius in x direction whereas the other radius is used for the y direction. Important subclass are ELLIPSISARC and HARVEYBALL. Both employ a start and end angle. These specify the filling within the ellipsis. For the HARVEYBALL, these values are predetermined to offer quick access to common fill values. IMAGE refers to an already existing icon or

figure included in the visualization. This includes vector and raster formats. Compatibility of these formats with the output format is checked within the respective RENDERERS and cannot be determined at design time with the current architecture. However, most graphics formats can be transformed to other formats that are compatible with the output format of the RENDERER, e.g. an SVG can be transformed to raster formats, e.g. PNG. The subclasses of EFFECT realizes ‘eye candy’ which commonly is used to highlight elements or foster an appealing visual representation. Effects are for instance BLUR, GLOW, and DROPSHADOW.

Other more complex visual symbols are realized as subclasses of COMPOSITESYMBOL. They use different PLANARSYMBOLS and commonly implement several convenience functions. Examples are DIFFERENCETEXT and PROGRESSBAR. The former realizes a two-way or three-way difference represents of text as depicted in Figure 5.34 on p.200 whereas the latter realizes the bars illustrated in the same figure.

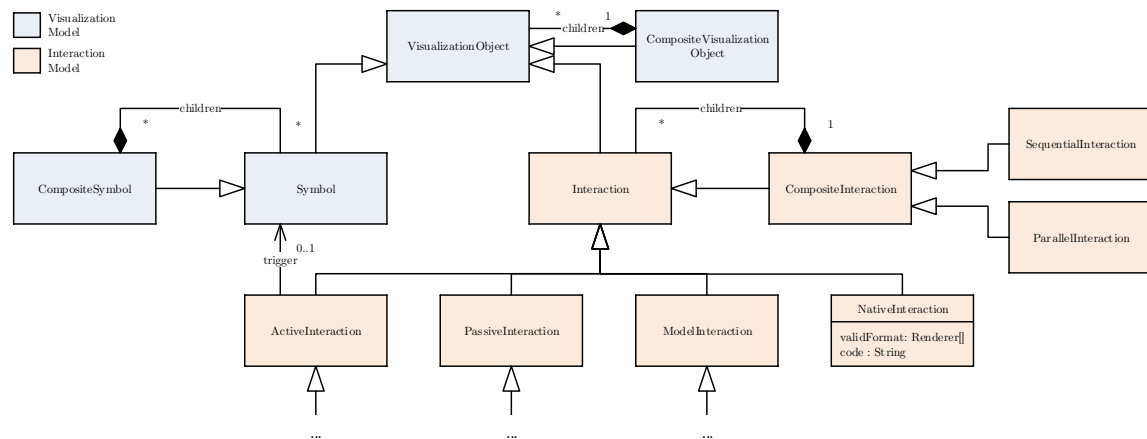


Figure 6.15: Interaction model of the visualization framework

After outlining visual concepts, we proceed towards interaction support realized by the framework. Figure 6.15 gives an overview of the different kinds of interactions that are incorporated with the visualization model.

- INTERACTIONS subsumes any concept that realizes interactivity. In an interactive visualization, they can be combined with visual symbols since they are able to pose as a VISUALIZATIONOBJECT. Both, SYMBOLS and INTERACTIONS can be stored within a COMPOSITEVISUALIZATIONOBJECT. Technically, any INTERACTION is realized with higher-order functions in JavaScript that are executed on invocation within the RAPHAELRENDERER.
- COMPOSITEINTERACTIONS denote a series of interactions to be performed which can be executed in sequence or in parallel. This behavior is realized by the subclasses SEQUENTIALINTERACTION and PARALLELINTERACTION.
- ACTIVEINTERACTIONS denote interactions that are triggered by an active user intervention commonly performed with an input device, e.g. a mouse click, a drag & drop operation, or a key press event.

- `PASSIVEINTERACTIONS` on the other hand can be triggered by user interventions but are not necessarily required to.
- `MODELINTERACTIONS` denote interactions which modify the underlying model or depend on it, i.e. either visual changes are propagated to the model or model updates are propagated to the visualization.
- `NATIVEINTERACTION` gives the developer direct access to the underlying source code of the generated visualization. On the one hand, this class provides capabilities such as
 - access to cutting edge features provided by just a few libraries which are used to realize concrete output formats and
 - quick access to the underlying visualization through direct (JavaScript) code injection.

On the other hand, this concept poses new challenges, e.g.

- the developer must be familiar with the target format,
- the type safety provided by the framework is not guaranteed when using `NATIVEINTERACTIONS`, and
- one has no guarantee that the visualization is valid, i.e. can be rendered or displayed to the user.

Thus, although powerful, the `NATIVEINTERACTION` must be dealt with caution since it can obscure the outcome of the model-to-model transformation.

Above interactions are further divided in concrete interactions that allow to manipulate the visualization and underlying model in manifold ways. We continue with the introduction of these concrete interactions. Figure 6.16 depicts the interactions within the visualization model. Since `NATIVEINTERACTION` is a concrete class, we discuss the `ACTIVEINTERACTIONS`, `PASSIVEINTERACTIONS`, and `MODELINTERACTIONS`.

The subclasses of `ACTIVEINTERACTIONS` are as follows.

- `MOUSEINTERACTION` implements events that are triggered by mouse. We distinguish between a simple `CLICK` and `DOUBLE CLICK`, a `HOVER` that is capable of triggering `INTERACTIONS` when the mouse hovers over a symbol or the mouse is dragged out of the hovering symbol, and a `DRAG & DROP` operation that is capable of triggering `INTERACTIONS` on `start`, `move` as well as `end` events of the `DRAG & DROP` operation and further invokes `INTERACTIONS` in the course of a valid or invalid drop. In this vein, validity of a drop is determined by the position of the `trigger` `SYMBOL` that is dropped and the `dropTargets` `SYMBOLS` given to the `DRAG & DROP` class.
- `KEYPRESSINTERACTION` denotes key events of the user. If the `trigger` is set for an `KEYPRESSINTERACTION`, the respective `INTERACTIONS` are only fired if that `|Symbol|` has been selected by a mouse click beforehand. Although the prototype of `MODELGLUE` interprets this class only for the `RAPHAELRENDERER`, we note that `RENDERERS` must take care of the transformation of different keycodes within the target format.

- TOUCHINTERACTION accounts for interactions of touch devices such as smartphones or tablets more and more used in industry. In MODELGLUE, most TOUCHINTERACTIONS are transformed to click interactions since we did not evaluate MODELGLUE on touch devices. We refer the interested reader to Kirschner [Ki12] for details on concrete touch interactions.

The subclasses of PASSIVEINTERACTIONS are as follows.

- HIGHLIGHT denotes INTERACTIONS that change a specific visual property commonly used to emphasize a SYMBOL.
- COMPENSATINGINTERACTIONS are a specific class of INTERACTIONS that compensate other INTERACTIONS on invocation. For instance, a HIDE INTERACTION removes a previously invoked SHOW INTERACTION from the execution stack.

The subclasses of MODELINTERACTIONS are as follows.

- CREATEOBJECT is utilized to create new OBJECTS within MODELGLUE, whereas SETOBJECT is utilized to change the name or description of it, and DELETEOBJECT initiates its removal. In the RAPHAELRENDERER, this is implemented as Asynchronous JavaScript and XML (AJAX) callback whereas the result indicates whether the transaction could be applied or not. Since the model is changed, these MODELINTERACTIONS typically are succeeded by an UPDATE INTERACTION.
- CREATEATTRIBUTE, SETATTRIBUTE, and DELETEATTRIBUTE are similar to the CREATEOBJECT, SETOBJECT, and DELETEOBJECT INTERACTION series.
- CREATEREference, SETREFERENCE, and DELETEREference are similar to the CREATEOBJECT, SETOBJECT, and DELETEOBJECT INTERACTION series. However, their explicit source and target URI is used to check the existence of an OBJECT first, then an ATTRIBUTE is used to create, set, or delete a reference from source to target.
- SETTASKOWNER is employed to change the role a task is assigned to. Note that this can be either persons or groups within MODELGLUE.
- SETTASKSTATE sets the state of a task based on an ID of a task within MODELGLUE.
- APPLYCHANGES applies the CHANGES to the model element the task refers to. On invocation, an implicit check of the task's state is performed, such that an explicit SETSTATE is not required. If OIDs of a third party system are involved, PROPAGATE TASKS will be generated to inform about changes that must be applied to other information sources.
- REVOKECHANGES revokes CHANGES, removes them from the task, and does not apply given CHANGES to a model element. Similar to the APPLYCHANGES interaction, the REVOKECHANGES interaction is aware of changes in a task's state. If OIDs of a third party system are involved, PROPAGATE TASKS are generated to inform about changes that must be applied to other information sources.

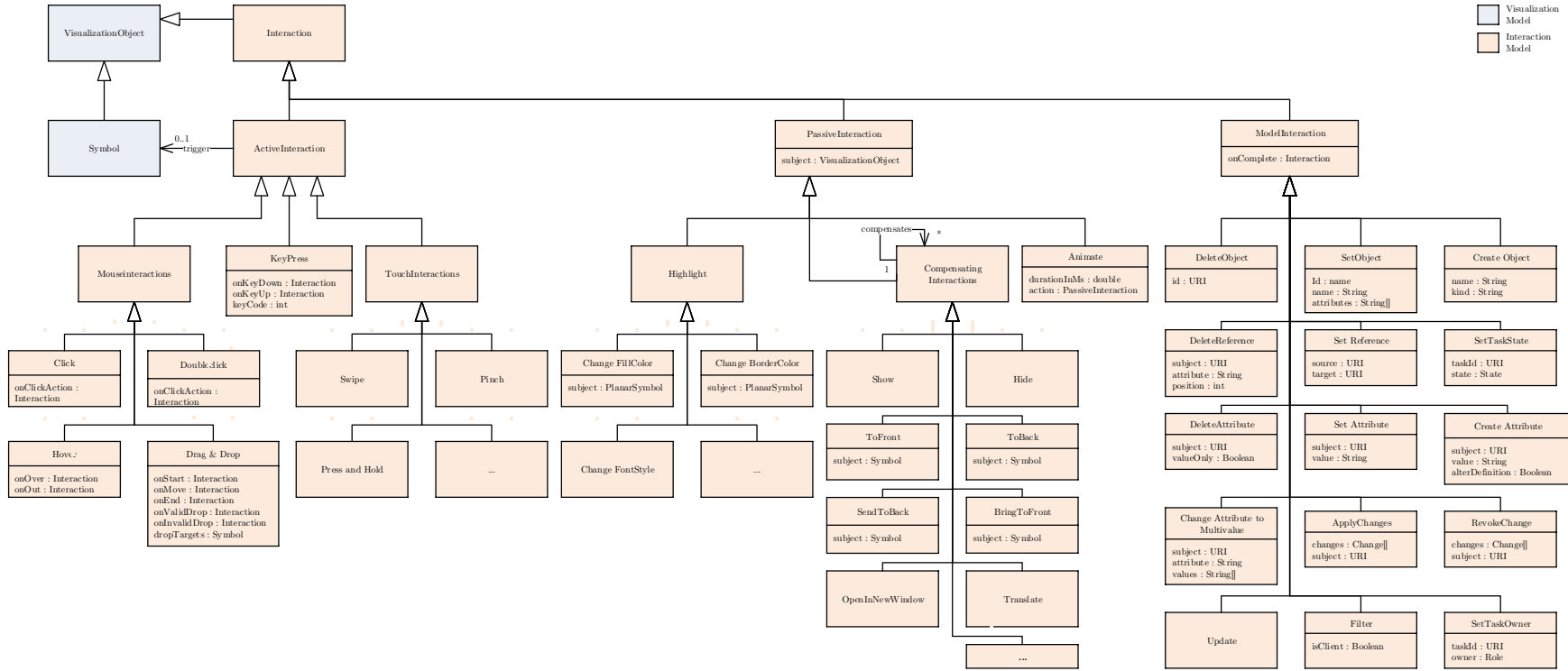


Figure 6.16: Interaction model of the visualization framework

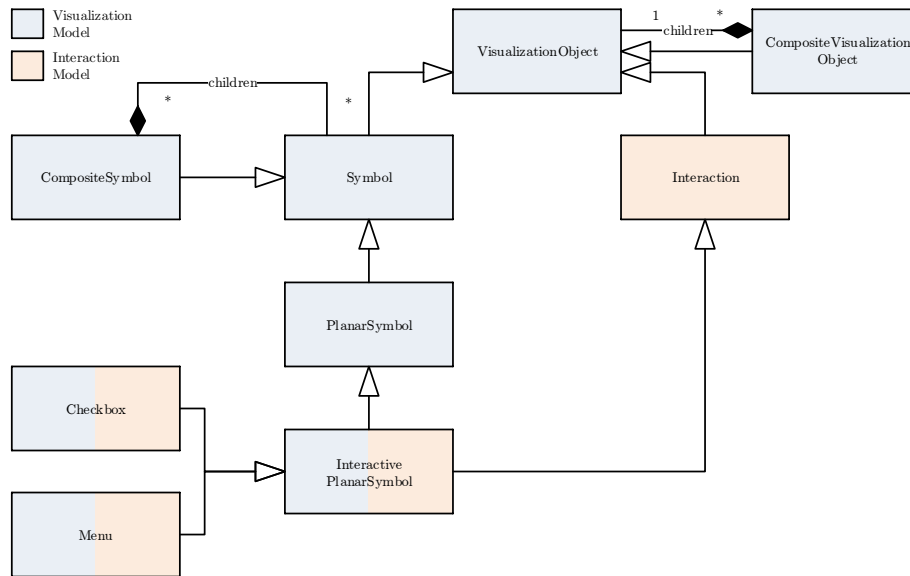


Figure 6.17: InteractivePlanarSymbol within the interaction model of the visualization framework

The interactive conflict resolution dashboard presented in Section 5.3 is an advanced UI incorporating sophisticated visual elements with interactions that even change underlying information. The nature of some elements is a hybrid combination of visual presentation and interaction. On the one hand, they are visually represented as a `PLANARSYMBOL`, they are in possession of a position, dimension and may even feature opacity, color, etc. On the other hand, these elements incorporate a specific predetermined interactive behavior. Users associate a specific set of interactions that already is determined at the design time of the control and should not be changed during the design of new visualizations. We refer to these elements as `INTERACTIVEPLANARSYMBOLS`. `INTERACTIVEPLANARSYMBOLS` realize for instance menu items that can be drawn based on `PLANARSYMBOLS` and incorporate predefined interactions. Since these `INTERACTIVEPLANARSYMBOLS` realize user controls, they are ignored by `RENDERERS` that realize formats that lack support for interactivity. Figure 6.17 illustrates the relationship of these `INTERACTIVEPLANARSYMBOLS` with the rest of the interaction model and the visualization model.

6.3.3 Towards Processing an Interactive Visualization

We utilize the visitor pattern [GHJ⁺94, p. 331ff] to process the object graph generated in the model-to-model transformation of an `ABSTRACTVIEWPOINT`. Figure 6.18 depicts the visitor pattern implemented in the visualization framework. The core functionalities are outlined below.

- The `ABSTRACTVISUALIZATIONOBJECTVISITOR` contains basic visit routines that implement the processing pipe for the object graph. Besides the visit routine for subclasses of `VISUALIZATIONOBJECT`, the `ABSTRACTVISUALIZATIONOBJECTVISITOR` implements a default behavior. After calling the concrete visit routine, the object is

6. Software Support for Federated EA Model Management

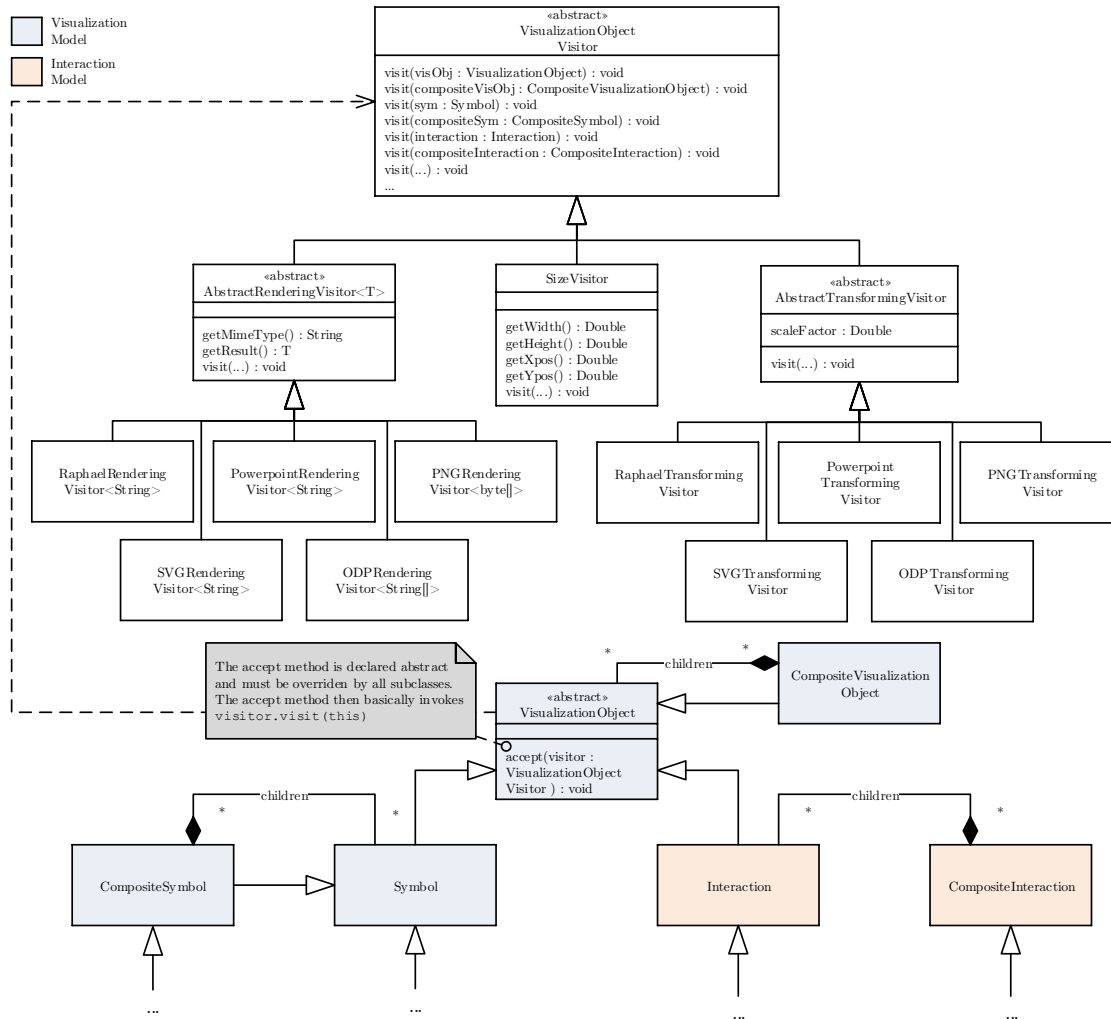


Figure 6.18: Visitor pattern within the visualization framework

type casted and used as an argument for the `visit(PlanarSymbol)` method. This way, operations that have to be performed on all visual elements can be handled centrally while concrete operations can also be performed. The object then is casted again and travels through the `visit(Symbol)` and `visit(VisualizationObject)` methods. This default behavior can be altered in the subclasses of `ABSTRACTVISUALIZATIONOBJECTVISITOR`.

- The `ABSTRACTRENDERINGVISITOR` implements operations that have to be performed in the course of rendering an object graph to an arbitrary output format. Thereby, the `ABSTRACTRENDERINGVISITOR` implements actions that do not depend on the output format. Its subclasses on the other hand handle actions that are format specific.
- The `SIZEVISITOR` calculates the overall dimensions of a visualization in a target format independent unit. The size of a visualization is based on the entire object graph that is produced as a result of an `ABSTRACTVIEWPOINT`'s transformation. The

SIZEVISITOR further determines the outermost coordinates that are given by the tuple $(x|y)$ which gives the top left coordinate, also known as origin, of a visualization.

- Subclasses of the ABSTRACTTRANSFORMINGVISITOR transform the coordinate systems of the object graph to the coordinate system of the target format.

Note that the concrete classes of VISUALIZATIONOBJECT, i.e. subclasses of SYMBOL, COMPOSITE SYMBOL, INTERACTION, and COMPOSITE INTERACTION override the polymorphic accept method, such that the respective visit method of the visitor is invoked by this class (cf. Figure 6.18).

The visitor pattern comes with well-known advantages and drawbacks [GHJ⁺94, pp. 335–337]. We briefly outline the advantages and explain how we exploit these in the prototypical implementation of MODELGLUE. Further we discuss the disadvantage of the visitor pattern and detail how we cope with it in MODELGLUE.

- **Easy introduction of new operations:** New operations that work on an object graph can be easily added through the introduction of a new ABSTRACTVISUALIZATIONOBJECTVISITOR.

In MODELGLUE we regard the visualization model as stable set with finite elements. It is more static than the operations performed on it. One could think of additional visitors for applying styles or templates to an already layouted object graph.

- **Separation of concerns:** Similar operations can be dealt with in the visitors and are not spread across the object graph. “Unrelated sets of behavior are partitioned in their own visitor subclasses. That simplifies both the classes defining the elements and the algorithms defined in the visitors. Any algorithm-specific data structures can be hidden in the visitor” [GHJ⁺94, pp. 335–336].

In MODELGLUE, we separate coordinate transformation, rendering, and calculation of the overall size of a visualization.

- **Costs of adding new elements to the visualization model:** A known limitation of the visitor pattern is the cost of an extension of the object structure of concrete elements, i.e. for any new subclass of VISUALIZATIONOBJECT all visitors must be adapted potentially.

In MODELGLUE however, we implement a default behavior in the ABSTRACTVISUALIZATIONOBJECTVISITOR. Although we regard the object structure, i.e. VISUALIZATIONOBJECTS and its subclasses, as stable such that the drawbacks of the visitor pattern do not apply that frequently, this default behavior must be understood by the framework developers. Only then, the developer can consciously decide to override the default behavior.

- **Accumulating states:** Visitors can carry variables that track the state of the entire object graph or parts thereof. Without the visitor pattern, this state must be passed as an argument to the operations that traverse the object graph or must be implemented as global variables.

In MODELGLUE, we track the state of the entire object graph in the visitors, e.g. the

overall size of a visualization or the `StringTemplate` that carries the generated JavaScript code (cf. Section 6.3.4).

- **Breaking encapsulation:** The visitor pattern assumes that an interface of an element is powerful enough to allow visitors access to relevant member variables. The visitor “pattern often forces you to provide public operations that access an element’s internal state, which may compromise its encapsulation.” [GHJ⁺94, p. 337].

In `MODELGLUE`, we access common variables by superclasses. To access specific variables throughout the rendering process, we rely on introspection. We employ a custom annotation `@Serializable` which retains at runtime. This annotation carries parameters specifying whether to access a Java field via its `get` method or directly via the respective field facilities. The former is a little bit slower whereas the latter is the fastest technique we came up with during four years of implementation work. This annotation technique is considerable faster than using mechanisms offered by the Enterprise Java Beans (EJB) standard. Note that access methods which just access fields (getter and setter) potentially could be optimized during byte code compilation or even by a Java virtual machine (VM) that allows just-in-time compilation.

On the other hand, the visitors are a good place to introduce caching mechanisms. In fact, during performance optimizations, we introduced a cache for rendering visual objects. In our prototypical implementation, typesetting text took considerable computational efforts. Thus, a `TEXT` object with the same string, font size, font face, width, and height parameters now is not measured twice. With caching, typesetting equal `TEXT` objects boils down to a string and several number comparisons which reduces computational efforts tremendously.

We discussed several advantages of the visitor pattern and detailed how we cope with its drawbacks. The visitors serve to process the entire class hierarchy of the visualization model and build the basis for the visualization process which is explained next.

6.3.4 Visualization Process

The `VISUALIZATIONPROCESSOR` brings together the static structure presented in Section 6.3.1 and the visitors described in Section 6.3.3. It receives a configuration which includes information on the databinding, filters, and an extension indicating the target format as well as further visual parameters that are viewpoint-specific. Figure 6.19 gives a highlevel overview of the visualization process. In the following we describe the interplay of the different concepts presented above informally.

In a first step, a concrete `DATABINDING` is fetched from the `DATABINDINGFACTORY`. The target format of the visualization and the viewpoint determine which `DATABINDING` to return. In the bind phase of the `DATABINDING`, the actual information is bound to the view model. In this phase, the first model-to-model transformation takes place, i.e. queries are executed on the source model whereas the results are transformed to a view model. An instance of a view model is an object graph in a format that is convenient to process by the algorithms used in the respective viewpoint. In parallel, a concrete `ABSTRACTVIEWPOINT` is fetched from the `ABSTRACTVIEWPOINTFACTORY`. Subsequently, this viewpoint is configured by

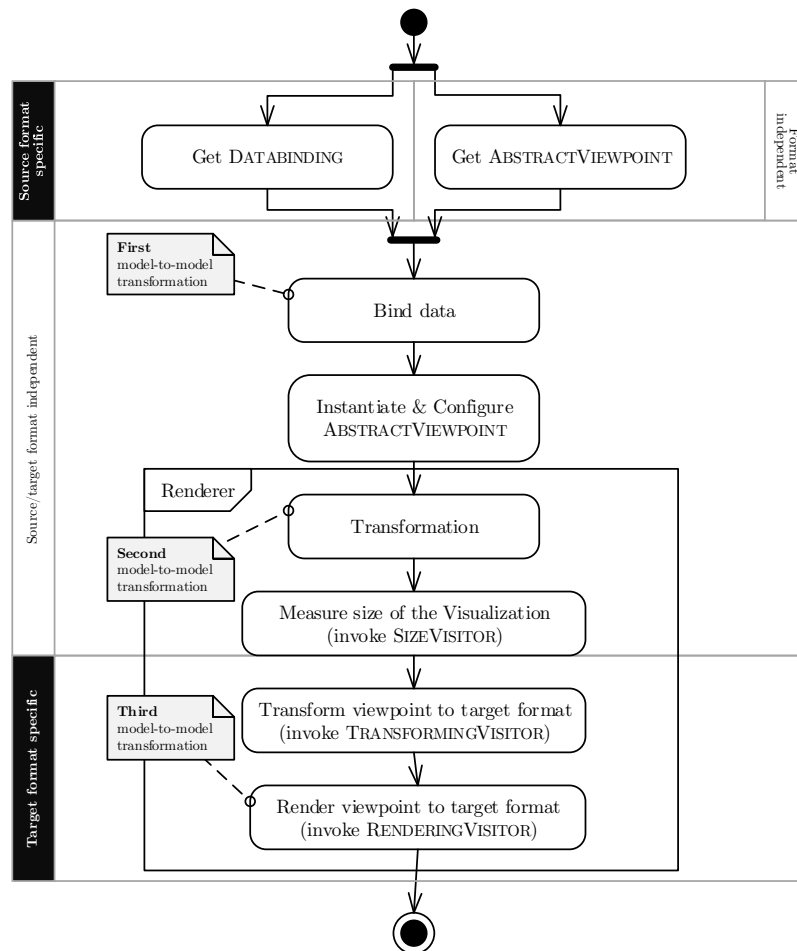


Figure 6.19: Overview of the Visualization Process as an Activity Diagram

injecting the `DATABINDING` and visual parameters that are to be processed during the transformation.

In the next step, a concrete `RENDERER` is fetched from the `RENDERINGFACTORY`. This renderer executes the second model-to-model transformation, i.e. the viewpoint performs a static type casts of the `DATABINDING` such that the viewpoint can access the convenient view model; this view model is transformed into objects of the visualization model. Put differently, during transformation, the view model is bound to visual elements that are instances of the visualization model. The result of this transformation is an object graph of visual elements, i.e. `VISUALIZATIONOBJECTS`.

Next, the concrete `RENDERER` employs the `SIZEVISITOR` to calculate the dimensions of the visual object graph. Next to determine the overall size in an output-format independent way, a concrete `TRANSFORMINGVISITOR` is called by the concrete `RENDERER`. The `TRANSFORMINGVISITOR` fits the visualization to the target coordinate system. In this step, the object graph is modified, i.e. coordinates of the target format are applied to each object.

In a final step, the modified object graph is transformed to the target format. This transformation process varies considerably and strongly depends on the output format that has to be produced. For PNG we employ the build-in Java Abstract Window Toolkit (AWT) libraries, for PPT/PPTX, SVG, ODP, and JavaScript (Raphaël), we employ the StringTemplate library [Pa14]. This library is an efficient means to transform a POJO graph to an arbitrary string-based output format. Depending on the output-format, some post-processing might be necessary. For instance, for PPT/PPTX and ODP, a .zip archive has to be created that follows a certain schema (cf. [Mi14]). For the JavaScript output, the Raphaël framework serves as low-level facility to abstract from browsers. The JavaScript format is further equipped with additional libraries that implement more complex interactions that can be handled without further server callbacks. In our prototypical implementation, a considerable part of the interaction logic is implemented in JavaScript. This way, highly responsive client-side interactions have been realized. Besides the process, Figure 6.19 also depicts which of the steps are format specific and which steps are fairly general for all source and target formats.

6.4 Implementing a Real-time Conflict Management Dashboard

After introducing the framework for interactive visualizations, which builds the technical foundations for our advanced UIs, we further detail the implementation of the conflict management dashboard. We start the discussion with a description of the more complex conflict management dashboard since visualizing model differences is done in a similar manner but can be regarded less complex.

6.4.1 Layout and Layers

A graph layout is used to calculate the positions of objects at layer 1 and layer 3. A common property of graph layout algorithms is that they rearrange the entire graph each time the layout is calculated.

In our implementation, we use the `mxHierarchicalLayout` of the JGraph library [JG14]. Its layout remains relatively stable, i.e. if the visualized information stays the same, the same result is produced whereas minor changes in information and, thus, the underlying model, produce minor visual changes, eventually.

The visualization framework allows to combine viewpoints via 1) `INTERACTIONS` and 2) new viewpoints that instantiate others. We applied the two `COMPENSATINGINTERACTIONS`, `HIDE` and `SHOW`, to realize the layer concept within the visualization. Thereby, `SHOW` displays a `COMPOSITEVISUALIZATIONOBJECT` that contains both, `SYMBOLS` and `INTERACTIONS`. The `HIDE` action is used as a `COMPENSATINGINTERACTION`, since it compensates the `SHOW` interaction. On a single visualization, `COMPENSATINGINTERACTIONS` have little impact. However, when synchronizing visual manipulations, `INTERACTIONS` are invoked on different machines. In the long run, not eliminating `INTERACTIONS` which are compensated by others would lead to unnecessary invocations. For instance if a layer in the conflict

management dashboard is opened and closed several times this will lead to a flickering effect on synchronization because show and hide actions are executed multiple times.

6.4.2 Conflict Tasks

In the following, we detail how we implemented the different interactions on the conflict management dashboard. Thereby, we employ a visual object graph, where appropriate. Note that we abstract from the composite pattern and refer to the class names that can be found within the different interaction models presented above.

Viewing a task: At layer 3 of the conflict management dashboard (cf. Section 5.3.2.3), a `PARALLELINTERACTION` is used to allow the user to view tasks. Thereby, the state of a task is set via the `SETTASKSTATE` to ‘reviewed’ while in parallel an instance of `SHOW` is invoked to display the task’s content.

Forward is realized with an `IMAGE` and a `CLICK`. On a mouse click, a `CHANGEOWNER` is invoked. This callback returns a blocking dialog which allows to set the owner of a task. Through a `SEQUENTIALINTERACTION`, the dialog is followed by an `UPDATE` of the interactive visualization.

Finish is realized with an `IMAGE` and a `CLICK`. On a mouse click, a `SETTASKSTATE` is invoked with the state ‘resolved’. Through a `SEQUENTIALINTERACTION`, an `UPDATE` of the interactive visualization is invoked.

Ignore uses a different `IMAGE`, but is realized similar to the finish action.

Approve is realized as two iconified buttons which are an instance of `IMAGE` in the visualization framework. Figure 6.20 depicts the object graph of the approve interaction. For each change which needs approval, it can either be decided to apply a change or to revoke a change.

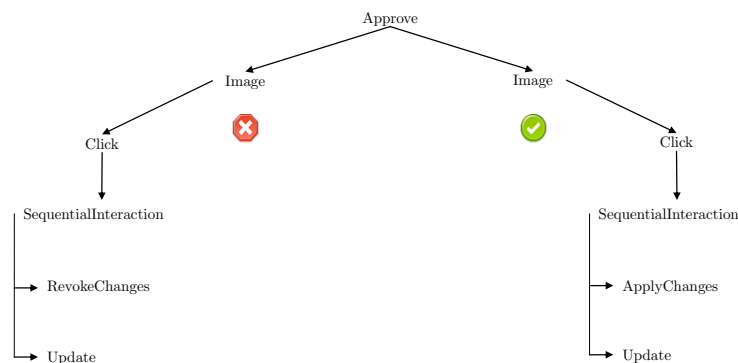


Figure 6.20: Object graph for the approve interaction within the conflict management dashboard

Validate uses an icon for each change that already has been applied to a model element. A click on such an icon revokes this change. Figure 6.21 depicts the respective object graph.

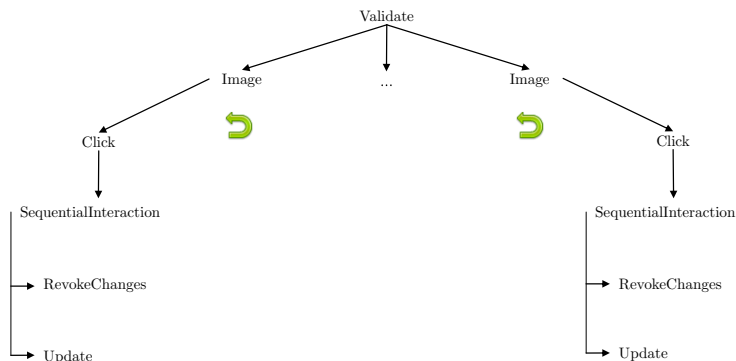


Figure 6.21: Object graph for the validate interaction within the conflict management dashboard

Document is realized with an `IMAGE`. On `CLICK`, the `OPENINNEWWINDOW` is invoked with the respective URI of the model element to be documented. Further, the state of the `TASK` is set implicitly to ‘reviewed’ and can be set within a document task explicitly (see **Finish**).

AssignRole is similar to the `Document` task. Instead of the URI of the model element, its settings are shown. The dialog has been illustrated in Figure 6.10 on p. 224.

6.4.3 Real-time Collaboration

In his master’s thesis, Höfler [Hö13] explored the possibilities of using NodeJS [Jo14] to synchronize different browsers in order to edit the same text simultaneously. Build on the works of Höfler [Hö13], Schrade [Sc13] uses NodeJS to synchronize interactive visualizations that run in different browsers on distributed machines. Thereby, he uses NodeJS and websockets [FM11] to issue broadcasts of the configuration.

Before we proceed to explain the synchronization concept of interactions issued by different browsers, we further detail the role of the class `UPDATE` within the interaction model. Figure 6.22 sketches the core interaction between the browser of the user, and the server, running the visualization component of `MODELGLUE`. Initially, the browser starts a request with a certain configuration which is given by its context and commonly is part of the current page or target page identified by the URI. The configuration is processed by the server which returns an interactive visualization that, again, includes the configuration. In this example, we assume that the `RAPHAELRENDERER` is used to generate the visualization, i.e. JavaScript code is produced by the server that is interpreted by the JavaScript engine of the browser. The configuration thereby is a JSON string and remains as variable within the generated JavaScript code. For now, let us assume that any `PASSIVEINTERACTION` that is invoked by the user is stored within the configuration. Thereby, the configuration is altered such that invoked interactions are temporarily stored within the configuration denoted ‘config*’ in Figure 6.22. Above, we detailed the notion of `MODELINTERACTIONS` which manipulate the underlying model of a visualization. Commonly, these `MODELINTERACTIONS` are followed by a subsequent `UPDATE` which sends an Asynchronous JavaScript and XML (AJAX) call containing ‘config*’ as parameters to the server which returns a visualization that

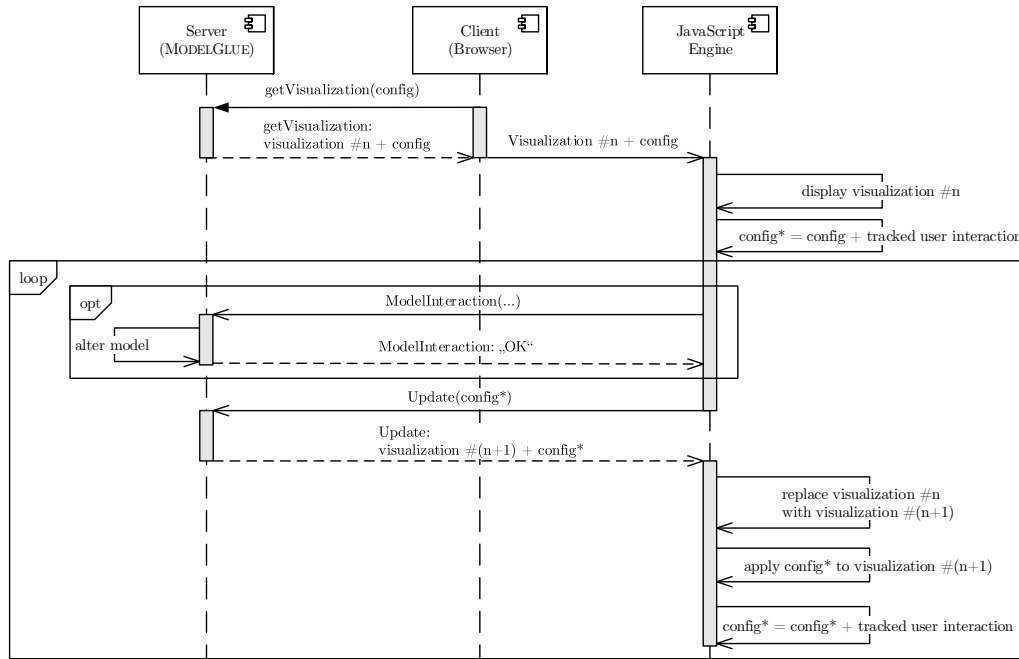


Figure 6.22: Sequence diagram to generate user-specific visualizations with an Update according to [Sc13, p. 35]

reflects the changed model. On success, the UPDATE manipulates the Document Object Model (DOM) tree and restores the visual parameters of the visualization, e.g. pan and zoom. Moreover, any INTERACTIONS that are stored within ‘config*’ are invoked such that effectively the state of the visualization is maintained by the UPDATE. Note that COMPENSATINGINTERACTIONS build an exception; COMPENSATINGINTERACTIONS are realized as higher order functions in JavaScript similar to INTERACTIONS. Example 6.3 illustrates their behavior.



EXAMPLE 6.3: CompensatingInteractions in JavaScript

On invocation of a COMPENSATINGINTERACTION I_1^c the JavaScript engine is told to check the existence of any Interaction I_2^c currently stored within the configuration. If I_2^c is stored within the configuration and I_1^c is invoked, I_2^c is removed from the configuration. This way, I_2^c compensates I_1^c .



We tried different designs to synchronize the interactive conflict management dashboard. In the following, we discuss the different alternatives and outline advantages, disadvantages as well as arising challenges when implementing the different approaches. Thereby, we make use of the UPDATE functionality, i.e. maintain the state of a visualization and invoke INTERACTIONS stored within its configuration.

Broadcasting the configuration: In [Sc13, p. 38], Schrade assumes that the configuration of a visualization is altered locally at the client. The MODELGLUE server is employed to generate a separate visualization for each client. An advantage of this solution is that it is easy to realize and different access rights, i.e. permissions on model elements, can be applied to the generated visualization individually. A major drawback of this solution is that the visualization is generated $n - 1$ times for n clients after a broadcast. To cope with this drawback, Kirschner [Ki14] introduces caches that are employed to speed up the rendering process for similar visual elements. For instance, typesetting a TEXT object with a specific font face and font size for thousands of elements requires considerable computational effort. A cache for such an element on the other hand compares if the same string is represented by the TEXT, the same font and size is used, and the dimensions (width, height) are the same, i.e. the computational effort is reduced to two string compares and three floating values. Obviously, this mechanism can be improved by building a hash over the involved values. However, after the introduction of this cache, the generation of visualization has been considerably faster.

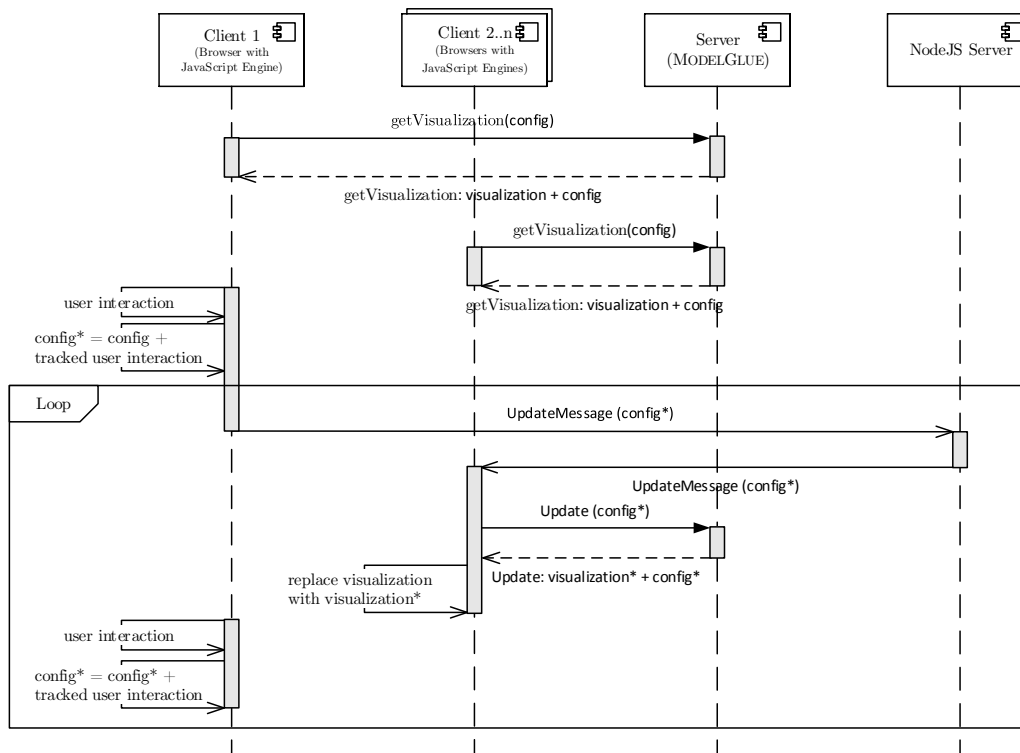


Figure 6.23: Sequence diagram illustrating the broadcast of a configuration to generate user-specific visualizations to enable visual real-time collaboration developed with Schrade [Sc13, p. 38]

Figure 6.23 depicts a single user denoted as *Client 1* that requests a visualization from the MODELGLUE server with a given configuration that may or may not be embedded within a web page. The MODELGLUE server generates a visualization according to the configuration. Within a collaborative conflict resolution session,

other parties, denoted *Clients 2..n*, participate. Initially, each client gets served by the MODELGLUE server in the same manner as *Client 1*. If *Client 1* issues a *visual update*, the configuration of that client is broadcasted to *Clients 2..n* via the NodeJS server. Subsequently, *Clients 2..n* query for a new visualization with that altered configuration, i.e. they invoke the same UPDATE functionality than detailed above.

A more efficient alternative can be realized if underlying information has not been altered; the configuration can be analyzed by the clients and INTERACTIONS can be executed directly by the JavaScript engines of the *Clients 2..n*. This way, the MODELGLUE server does not have to generate new visualizations for mere visual interactions. An even more efficient variant of these considerations is discussed below (see **broadcasting interactions**).

Broadcasting the visualization In [Sc13, p. 39], Schrade presents an interesting design alternative. Thereby, the MODELGLUE server notices if the underlying model of a visualization is manipulated and initiates a broadcast to all connected clients through the NodeJS server.

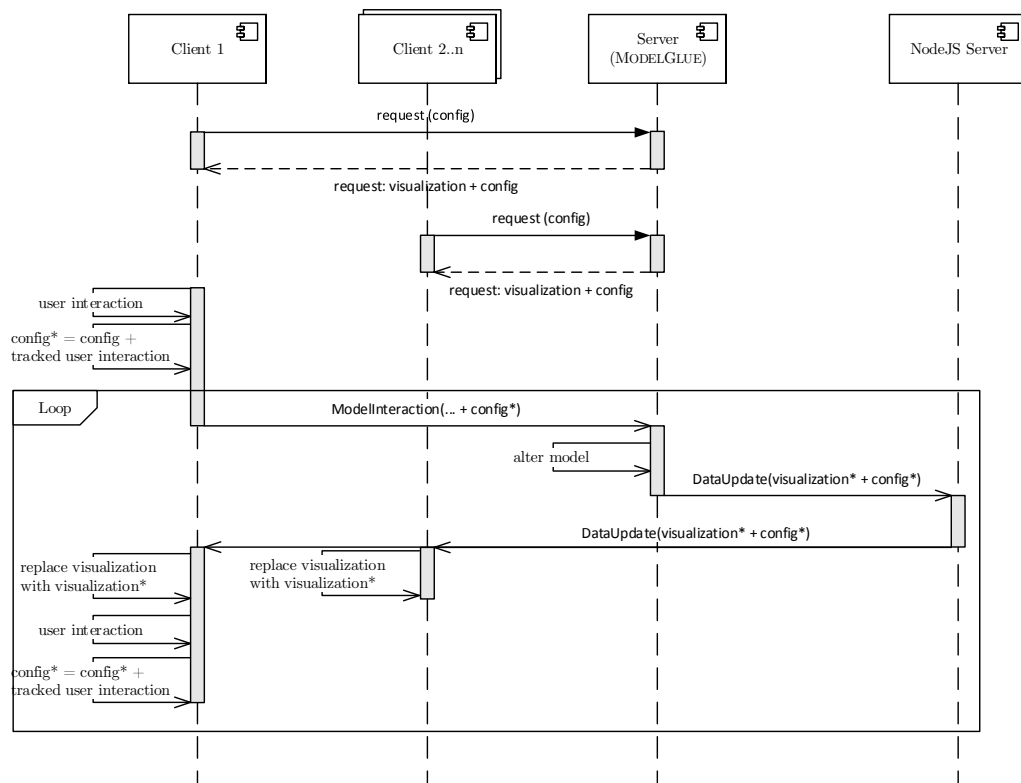


Figure 6.24: Sequence diagram illustrating the broadcast of a single visualization to enable visual real-time collaboration developed with Schrade [Sc13, p. 39]

Figure 6.24 illustrates the same situation as discussed above. This time, underlying information is altered by an arbitrary MODELINTERACTION. This is detected by the server such that a different broadcast behavior is used. Instead of the client issuing the modification, the MODELGLUE server initiates the broadcast after the

visualization has been rendered. This way, all connected parties *Clients* 1..n get the same visualization as well as the altered configuration. Both are then replaced locally. This exchange of a generated visualization that is send as serialized JavaScript code to the clients proved to be very efficient and its complexity to implement is similar to the broadcasting of the configuration. The advantage is that the visualization is only rendered once. Combined with our considerations for caching, the synchronization of visualizations boils down to network latency issues. However, during the evaluation of MODELGLUE, this design proved not to be a viable solution for Federated EA Model Management. With the current implementation, the access rights of the client that initiates the broadcast are used to generate the visualization. Since the generated JavaScript code is then distributed via a broadcast of the NodeJS server, the other clients see information that may not be accessible with their effective access rights, i.e. access rights become ineffective and may get violated.

Broadcasting interactions INTERACTIONS which do not manipulate the underlying model of a visualization can be replayed on each client. Thus, the NodeJS server is used to broadcast just the INTERACTIONS performed by a client. One has to be aware of raise conditions when implementing this principle. The notion of layers and COMPENSATINGINTERACTIONS can help to prevent some of the effects of raise conditions.

6.5 Summary

In this chapter, we revealed implementation details of the prototypical implementation of MODELGLUE. Thereby, we put emphasis on UI aspects. Especially, we revealed implementation details of the visualization component since this framework builds the technical foundation for our innovative UI designs proposed in Section 5.3. Where suitable, we referred to work of others which describes the technical details. In the final part of this section, we outlined how this framework is employed to implement the interactive visualization that realizes the conflict management dashboard. We discussed the software engineering challenges, provide rational for our design, and discussed different implementation alternatives we tried as well as their advantages, drawbacks, and lessons learned. This prototypical implementation builds the software-support for Federated EA Model Management and serves us to evaluate our concepts in collaboration with EA experts from industry.

In this chapter, we report on the evaluation of Federated EA Model Management and MODELGLUE. Researchers can take different paradigms to evaluate an approach, method, or artifact. We first provide a brief overview which paradigms researchers may chose from and subsequently provide rational for the design of a multifaceted evaluation.

7.1 Evaluation Design

Wohlin et al. [WRH⁺12, p.9] and Shull et al. [SCT01] distinguish two fundamental research paradigms:

- ‘*Exploratory research*’ focuses on the study of objects in their natural environment such that findings emerge from observations made. This requires a flexible design to adapt to new circumstances and lessons learned. Wohlin et al. further relate exploratory research to ‘*qualitative research*’ since this flexible design is informed by qualitative data primarily.
- ‘*Explanatory research*’ seeks to identify and quantify causal relationships. Wohlin et al. relate explanatory research to a fixed design since influencing factors are fixed beforehand. In this vein, they also refer to ‘*quantitative research*’ as explanatory research is primarily informed by quantitative data.

Wohlin et al. [WRH⁺12, p.24f] note that an empirical investigation on new technology, which shall be considered comprehensively, requires a combination of different methods. Thus, our empirical study is carried out combining survey as well as case study research.

Although explanatory research has an advantage over exploratory research, i.e. a rigorous quantitative data basis may serve as a foundation for comparisons and analyses by means

of statistical methods, Wohlin et al. conclude that both paradigms “should be regarded as complementary rather than competitive” [WRH⁺12, p. 9]. Shull et al. [SCT01] propose to combine different research methods for introducing software processes. Before we outline our approach, we briefly revisit different methods of empirical studies. Wohlin et al. classify the following three different empirical strategies [WRH⁺12, p. 10ff] that depend on the purpose of the evaluation and whether techniques, methods, or tools are to be investigated.

Surveys are means to collect information in order to describe, compare, or explain knowledge, feelings, values, preferences, and behavior of individuals or society [Fi13, p. 2]. Surveys are best to get information directly from people about what they believe, know, and think [Fi13, p. 24]. They can be carried out handing a self-administered (written or online) questionnaire that someone fills. Essential to a good survey design are the asked questions, answer choices, sampling methods, response rate, design, and the data analysis.

Experiments are means to learn the impact of a factor or variable on the studied setting [WRH⁺12, p. 11]. “Experimentation in software engineering supports the advancement of the field through an iterative learning process” [BSH86]. In this process, an experiment includes independent variables (*endogenous*), i.e. variables the researcher is in control of during the experiment and dependent variables (*exogenous*), i.e. variables the researcher does not or simply cannot influence during the experiment [WRH⁺12, p. 92]. An experiment commonly takes place in a controlled environment.

Case Studies are means to evaluate the benefits of methods and tools in an industrial setting [KPP95]. They allow researchers to “retain the holistic and meaningful characteristics of real-life events—such as individual life cycles, small group behavior, organizational and managerial processes[...].” [Yi09, p. 4].

In line with Yin [Yi09] and Wohlin et al. [WRH⁺12], we summarize the major characteristics of these methods in Table 7.1.

Characteristic	Survey	Experiment	Case Study	Source
Form of Research Question	who, what, where, how many, how much?	how, why?	how, why?	[Yi09, p. 8f]
Design Type	fixed	flexible	fixed	[WRH ⁺ 12, p. 12]
Qualitative/Quantitative	both	quantitative	both	[WRH ⁺ 12, p. 12]
Execution control	no	yes	no	[WRH ⁺ 12, p. 19], [Yi09, p. 8f]
Measurement control	no	yes	yes	[WRH ⁺ 12, p. 19]
Investigation cost	low	high	medium	[WRH ⁺ 12, p. 19]
Ease of replication	high	high	low	[WRH ⁺ 12, p. 19]

Table 7.1: Characteristics of different evaluation methods

Case studies focus on qualitative feedback to foster the understanding of Federated EA Model Management. Patton [Pa02, p.4] classifies three different sources of qualitative data: interviews, observations, and documents. Although experimentation is an important research strategy [BSH86, Ba96], we chose to carry out

- two case studies to get feedback from real-world applications as EA management is a discipline that is hardly to grasp reading a textbook,
- an interview series to gain further process understanding, and
- an online survey among EA experts to confirm our findings and assumptions.

Bringing in feedback from EA experts primarily serves to compensate lack of domain knowledge and gain insights in day-to-day issues and concerns of EA experts as well as to acquire their feedback and opinions to developed software artifacts, concepts and UI dialogs of MODELGLUE.

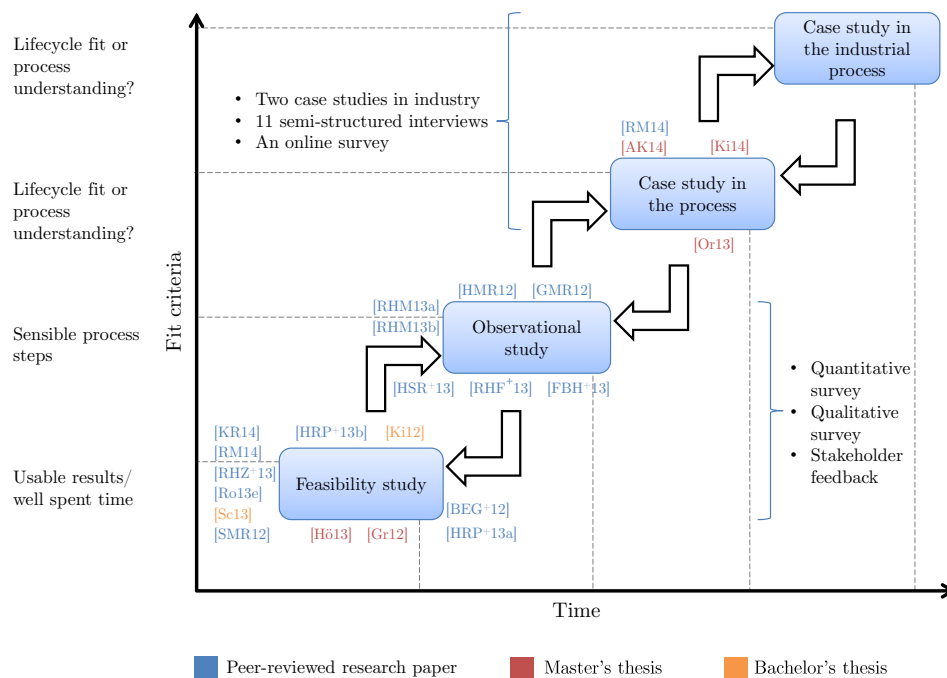


Figure 7.1: Evaluation steps of MODELGLUE combining different research strategies

Based on the ideas of Shull et al. [SCT01], Figure 7.1 gives an overview of our approach taken to evaluate MODELGLUE and related artifacts. It also incorporates preliminary results that have been published in our previous work. These build the foundation of the present thesis since discussions about empirical findings or technical feasibility at conferences and workshops certainly influenced our design.

Figure 7.1 depicts an iterative process to introduce new software processes. In the following, we briefly go through this process and point out the core publications. It starts with a feasibility study to validate if our ideas researched are doable; results have been presented

in different adjacent areas, e.g. developing EA visualizations [SMR12], extracting and importing data from an actual ESB [BEG⁺12, Gr12], towards results which build upon these foundations like the visualization of model differences [RM14] or n-way model merging [KR14]. In these publications, we report on feedback from practitioners who generated new input for further directions and adaptations of our designs.

Second, we carried out an observational study to find out whether the design and process steps that we proposed make sense. Thereby, empirical insights are reported, e.g. feedback on the proposed methods and interactive visualizations [RHM13b], but also open issues [HMR12] have been reported, as well as the status-quo in organizations [RHF⁺13, FBH⁺13]. These findings built the groundwork for our ideas how to address the challenges. Besides lessons learned during technical feasibility studies, e.g. [Ki12, Sc13], additional feedback served us for further iterations on the prototype presented in [HRP⁺13b].

In a third step, the design is evaluated with respect to an integration in the real lifecycle as part of case studies. Master's theses served as vehicle to evaluate our ideas with practical data [Ki14], or directly in practice [Or13]. Additional interviews gave us insights in further organizational aspects and furthered our process understanding [AK14].

According to Shull et al. [SCT01], the application in industry takes place in a fourth and final step to see if the process fits into the industrial setting. This phase is beyond the purpose of our case studies and require considerable efforts that go beyond a single PhD thesis (cf. Section 8.2 and 8.3).

In the remainder of this chapter, we detail the approach taken in two case studies and report additional feedback from industry. Throughout the evaluation, we held several retrospective meetings during which we reflected the expert feedback, developed solutions, and discussed possible adaptations. These were prioritized based on their technical complexity, eventual arising challenges, and potential benefits for the study or the organization. We present feedback, learnings, and improvements of our design during this phase of research. After each case study, we reflect on the findings and summarize both case studies in a separate section. Subsequently, we present additional insights from an interview series among 11 EA experts.

While we ran MODELGLUE with productive data, it has not been used in production, i.e. the organizations did not actually apply MODELGLUE in practice, but we employed backup files of productive systems utilizing MODELGLUE as software support for Federated EA Model Management. This way, we evaluate MODELGLUE and report on experiences gained in a real-world setting and on feedback gathered from EA experts applying MODELGLUE to their problems in EA management with productive data.

7.2 Case Study in the German Insurance Sector

The first case study took place in the German insurance sector. As of 2012, the organization counts 8,000 employees. As documented by Kirschner¹, the case study took place during September, 2013 and April, 2014 [Ki14, pp. 11–12]. Taking into account the initial interviews,

¹Prior to the case study, Kirschner has been briefed with the guidelines for conducting and reporting on case studies published by Runeson and Höst [RH09].

the evaluation of this first case study consisted of six interviews². In addition to the efforts described by Kirschner [Ki14, p. 12], two initial calls took place which are described in the following.

7.2.1 Initial Interviews

In an initial one-hour phone call, we intended to clarify if the organization is an interesting candidate and in turn is interested in trying out MODELGLUE with their organizational live data. During the call, we

- described our design, its purpose, and intended goals,
- got to know the context of the EA efforts the organization currently is pursuing,
- aligned expectations of a case study on both sides,
- sketched the degree to which the organization would be involved and has to provide support,
- formulated meaningful next steps and follow-up activities.

The outcome of our efforts was a precisely formulated scope, i.e. the case study embraces the exchange of timeslices of historical EA information and the feedback based on an analysis of this information within MODELGLUE. After this initial call, an informal Letter of Intent (LOI) was exchanged summarizing goals and follow-ups. This LOI was followed-up by a formal non-disclosure agreement (NDA) since we dealt with productive data of the organization. Thereafter, we received ~450 files in XLS/XLSX format that described enterprise models.

A first step towards their integration was a conceptual alignment. Since we were not familiar with the models, we prepared conceptual models using UML to describe what we received in XLS/XLSX format. Although the concepts have been discussed in the initial interview, the conceptual model helped us to understand their interrelationships and also built a good basis for discussions in the next call. Using conceptualized models described in UML revealed abstraction gaps and raised questions how to map these concepts, e.g. ‘how can an object of this class be identified uniquely?’.

In a second call, we clarified the developed metamodels and the different semantics of the concepts. To some extent, an intuitive understanding of the class’s identifier helped us to relate concepts; however the formal models (cf. Figure 7.2) made flaws visible. One major obstacle at this initial stage has been the unique identification of objects in an information source as well as across information sources. Later on, we had to rely on exact name matching mechanisms to relate objects.

²The exact details, e.g. duration of interviews, participants, job descriptions, company size, etc. are provided by Kirschner in [Ki14, pp. 11–12,69].

7.2.2 Context

In the following sections, we describe the environment of the organization and detail the metamodels of the involved information sources. Afterwards, we describe an integrated view on these metamodels. This integrated view on the conceptual models was the outcome of our efforts of the second interview.

7.2.2.1 Current Challenges, Technology, and Frameworks

Throughout the case study, we identified an abstraction gap (cf. Section 5.2.6.1) between involved models. The organization struggles with ambiguous concepts; in particular the alignment of concepts used in the more business-related models with those used in the more technical models poses a challenge. The diversity of models used throughout the organization is aggravated by modeling communities applying different notations and tools, e.g. natural language, UML, Event-driven Process Chain (EPC), and BPMN. Managing the interest of multiple stakeholders at a time is often rather difficult for the EA team. Oftentimes, multiple stakeholders of one modeling community have to be involved to derive planned states for an EA. The EA team currently pursues a long-term integration and consolidation of above outlined models. This includes overcoming the abstraction gap (cf. also Section 5.2.6.2). EA stakeholders within the organization already realized the benefits of a federal view on the different models and the benefits to work towards CCMC.

The organization implements an ITIL process. Further, they employ a web-based, model-driven EA repository. The EA repository realizes many concepts of Hanschke [Ha10]. Some of them are outlined in the course of the introduction of the organization's metamodels which we integrated.

7.2.2.2 Core Metamodels of Modeling Communities

We received ~450 files, representing historical data which served us as an input for an analysis of the EA between March 2009 and July 2013. These files were data dumps that served as input for the EA repository. These quarterly time-slices allowed the reconstruction of the EA model of each 16 quarters between Q3 2009 and Q2 2013. The given models represent the EA from different perspectives since the models come from different modeling communities. In the following, we briefly outline each community [Ki14, p. 70–72].

- **Product and service management** is concerned about the organization-wide product and service catalog. The catalog embraces products and services offered to other business units as well as to external customers. Product and service management is part of the controlling department—an administrative department on the same hierarchy level as EA management.

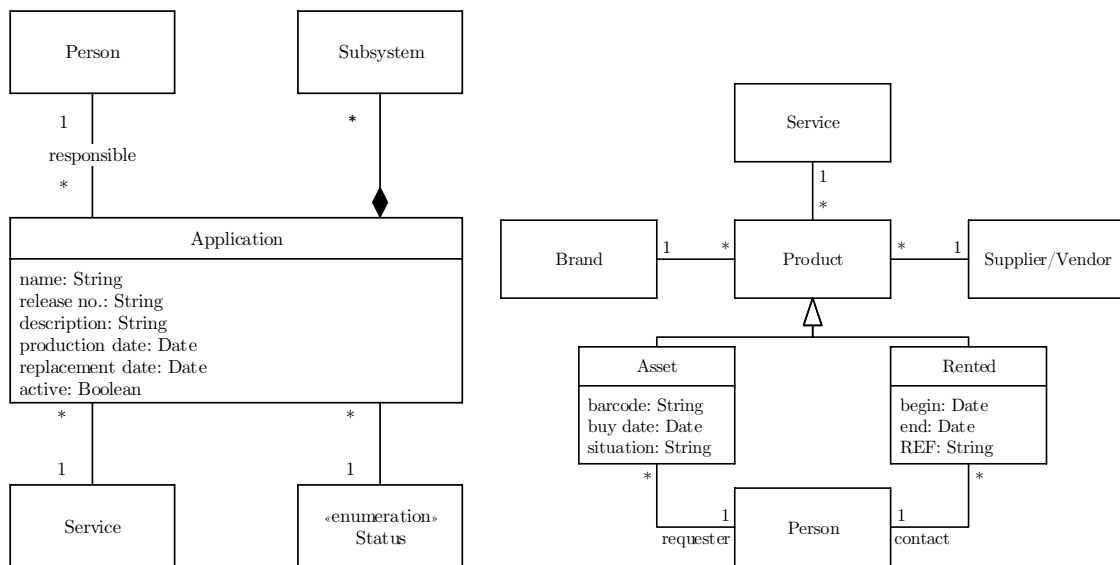
- **Application development** focuses on the implementation of company-specific applications. Application development also handles change requests issued by other business units.
- **IT asset management** is responsible for the procurement of IT assets. Thus, IT asset management oversees the entire life-cycle of an asset.
- **IT operations** maintains the CMDB and manages the technical IT infrastructure. Its terminology and concepts are based upon recommendations of ITIL [Ca11b].

Our analysis started by examining the information sources and building conceptual models thereof. We briefly introduce the three resulting metamodels. Note that we only detail core concepts that are relevant to understand the case and to gain insights into the difficulties when introducing Federated EA Model Management.

- The **application development model** is depicted in Figure 7.2(a). Within the community, the system is used to measure and control the development of individual applications. Thus, the central concept is APPLICATION which is used to describe individual software solutions developed in-house and are utilized within the organization primarily. It can consist of several SUBSYSTEMS and features several ATTRIBUTES like name, description, production, replacement date, etc. Moreover, every APPLICATION has a boolean flag indicating its status, i.e. whether it is currently ACTIVE or not, and a responsible person. Building the bridge to the next model, an application can realize or contribute to a SERVICE. The models we dealt with contained between $\sim 1,200$ and $\sim 1,600$ APPLICATION instances [Ki14, p. 78]. In [Ki14, p. 78], Kirschner provides additional insights in the trends of the models.
- The **IT asset manager model** is illustrated in Figure 7.2(b). Core entity of the model is PRODUCT which can be either RENTED for a specific duration or an ASSET that has been bought at a specific date. Similar to APPLICATIONS in the application development community, PRODUCTS can realize or contribute to a SERVICE. Respective models contained between $\sim 2,500$ and $\sim 3,700$ PRODUCT instances [Ki14, p. 78].
- The **CMDB model** is shown in Figure 7.2(c). A Configuration Item (CI) is a concept of ITIL which serves the documentation of IT infrastructure information. While more specific information is contained in the subclasses, a CI features common attributes such as, description, status, service level, etc. A GROUP is responsible for an CI and its semantics can be compared to our notion of role (cf. Definition 4.3). In our analysis, we focused on the concepts APPLICATION and INTERNAL SERVICES. The former denotes business applications whereas the latter denotes services, e.g. testing of server availability. The time-slices of the CMDB models were by far the largest concerning number of instances we dealt with, i.e. the models contained between $\sim 16,900$ and $\sim 22,300$ CI instances [Ki14, p. 78].

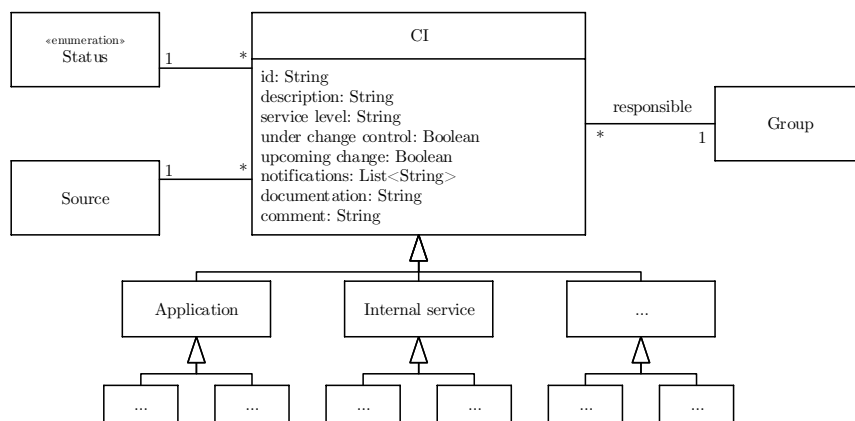
Note that we only received three models from the four communities outlined above. Although we had no actual model of the ‘product and service management community’, we considered

7. Evaluation



(a) Core metamodel of the application development community

(b) Core metamodel of the IT asset management community



(c) Core metamodel of the IT operations community

Figure 7.2: Core metamodels of the information sources [Ki14, p. 72]

several concepts of the other models boundary model elements (cf. Definition 4.7). Thus, the product and service catalog is key since many MODELLEMENTS of other models have relationships to services which are contained therein. In particular the service identifiers are stored as foreign keys in instances of the OBJECTDEFINITIONS APPLICATION and PRODUCT. In the following, we point out how these models of different modeling communities have been put together into a coherent EA model.

7.2.2.3 Integrated View as an EA Model

Although we diagnosed in our initial calls that the organization can be considered as a federated EA model environment, during the integration of the provided models, we experienced considerable difficulties to map the models physically.

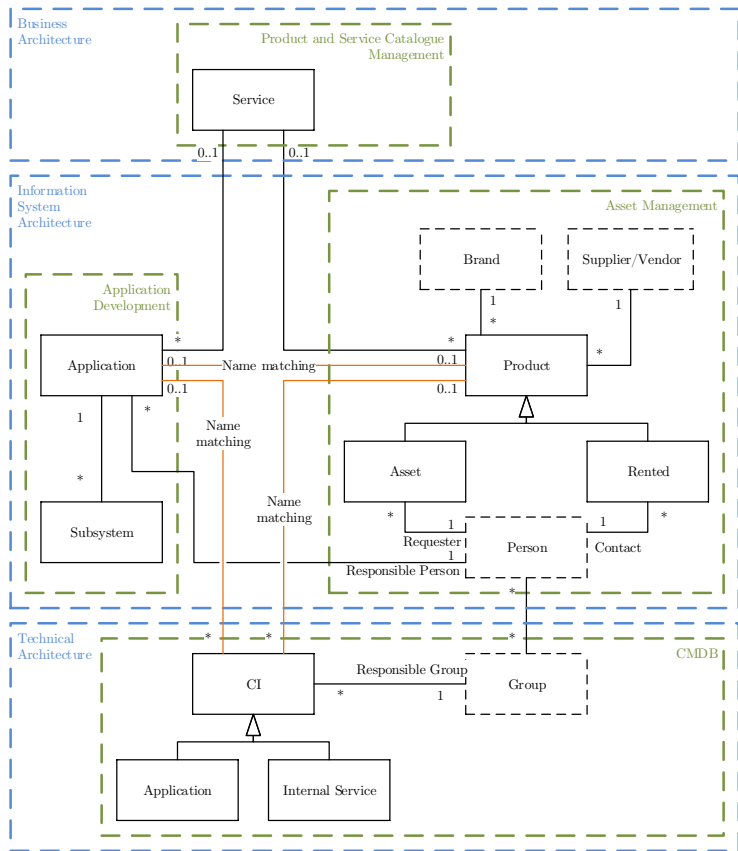


Figure 7.3: Integrated current state of the EA [Ki14, p. 74]

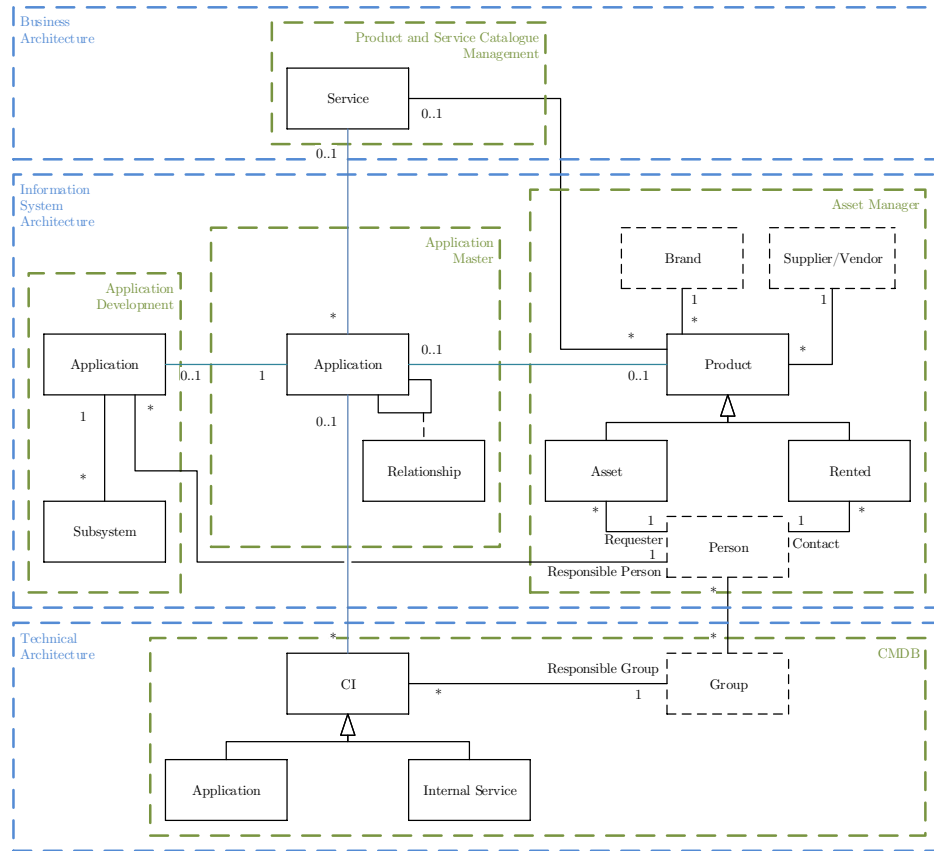


Figure 7.4: Integrated planned state of the EA [Ki14, p. 75]

Figure 7.3 depicts an integrated view of the EA model we discussed during the second initial interview (cf. Section 7.2.1). In this phase, the conceptual model helped us to understand how to develop a physical mapping. The model also depicts annotations we made during the discussion. In particular the matching of objects between the different classes was discussed. We had to rely on name matching techniques as foreign keys to a large extent were not used in the provided data sets.

Kirschner diagnoses a lack of direct references between the core elements. However, to some extent APPLICATIONS and PRODUCTS could be mapped via their corresponding SERVICES. Although the resulting mapping was sometimes ambiguous, in ~25% of all cases we found corresponding links to OBJECTS via exact name matching. For the remaining ~75% we cannot tell whether a relationship exists or not. This has to be further investigated by mapping all involved PRODUCTS to corresponding APPLICATIONS elements manually.

In this vein, we identified three abstraction gaps: the first was between GROUP and PERSON, the second between CI and APPLICATION, and the third between CI and PRODUCT. Resolving the first abstraction gap was beyond the scope of our evaluation; our primary concern was to map CIs to PRODUCTS, CIs to APPLICATIONS, and APPLICATIONS and PRODUCTS.

Overcoming the abstraction gap automatically turned out to be rather problematic, since more than one CI corresponds to a single APPLICATION and PRODUCT. Hence, we proposed our concept of the abstraction gap resolver to the enterprise architects. While they agreed that the concept could provide utility with respect to the concrete problem, they did not actually apply the abstraction gap resolver (cf. [RHM13b] and Section 5.2.6.2) to overcome the abstraction gap.

Further, we also identified boundary model elements (cf. Definition 4.7) within the EA model. GROUPS and PERSONS are primarily managed by the human resources (HR) unit and currently only represented by its foreign key. Thus, the access rights could not be automatically imported.

Kirschner proposes to import access rights to other systems based on directory systems. Such a solution presumes that every model that is integrated with the EA model relies on such a directory service. Other boundary model elements such as BRAND and SUPPLIER have been identified. These could build the starting point for an integration of additional information sources. Prior to such an endeavor one has to assess the potential benefits of such an integration (cf. Section 5.2.2) by determining the concerns such an extended EA model could address.

Besides this integrated view, we also discussed future states of the EA model. Figure 7.4 depicts the outcome of our discussions. The EA team's intent is to introduce a new model referred to as 'Application Master'. It is meant to support the EA team to overcome the abstraction gap between CONFIGURATION ITEM (CI) and APPLICATION and PRODUCT. As the name 'Application Master' suggests, it should become the new leading system regarding applications utilized and developed throughout the company. Similar to the case we investigated in [BEG⁺12], the connection between APPLICATIONS is also of high importance since it indicates a connection between CIs. That also means CIs must only be connected if their high-abstraction counterparts, i.e. APPLICATIONS within the application

master are linked via a `RELATIONSHIP`. Additionally, this new ‘Application Master’ also incorporates an identifier that is unique for each `APPLICATION` and serves as foreign key in all other models.

7.2.3 Execution, Feedback, and Adaptations

After the model integration, we imported the models into an instance of our prototypical implementation of `MODELGLUE` hosted at a secure infrastructure and granted both enterprise architects access to this system.

While Kirschner reports on the case studies extensively [Ki14, pp. 68–86], we summarize his results and subsequently reflect the case study in the light of additional insights through our interview series and the refined prototype.

Thereby, we describe each of our findings with 1) a brief title followed by 2) an outline of the context during the interviews, 3) core statements of the expert feedback, and 4) a brief discourse providing further insights and discussions.

7.2.3.1 General Feedback to the Application

Increased automation due to `MODELGLUE`:

Context: `MODELGLUE` seeks to import information automatically; however it also requires efforts to define a technical mapping.

Feedback: One enterprise architect argues that in many cases model matching cannot be accomplished since human-readable formats are involved; mainly because respective tool support, e.g. office applications, are easier to use. Many stakeholders favor the perceived flexibility of these tools over a more formal approach that would foster automation.

Reflection: On the one hand, this statement of the EA expert supports our design decisions that modeling communities require a considerable degree of freedom when modeling. However, as of today `MODELGLUE` cannot be used to integrate fully unstructured content. In his PhD thesis, Neubert [Ne12] proposes a means to structure content iteratively and incrementally such that a formal metamodel evolves over time. Since `MODELGLUE` is based on the same platform, further research could combine his approach and `MODELGLUE` with Natural Language Processing (NLP) techniques to facilitate structuring unstructured content.

7.2.3.2 Tasks

Conflict task of `MODELGLUE`:

Context: All task types (cf. Section 5.2.5.5) were critically discussed with respect to their semantics and likelihood to arise during a merge of models.

Feedback: Both enterprise architects regarded the task types helpful, in particular to determine the impact of a merge on model consistency. Since this conversation

about the task types took place in context of the conflict detection and different kinds of conflicts, in-depth discussions revealed that the enterprise architects consider model/metamodel conflicts more likely to arise with repositories of communities that maintain information found at higher layers (cf. Figure 2.1.1 on p. 16) as they model on a higher abstraction level. In contrast, one enterprise architect states that physical hardware would never change its purpose and can be considered more rigid with respect to model/metamodel conflicts.

Reflection: We anticipated different kinds of tasks and the need for different conflict resolution strategies. In particular, we claim that it is required to adapt these strategies to reflect organization-specific circumstances. These statements confirm our design and give first insights how EA expert assign specific conflict types to semantics of their models.

Ignoring tasks in MODELGLUE:

Context: We discussed the importance of some tasks and conflicts with respect to particular day-to-day situations that the enterprise architects experience.

Feedback: The enterprise architects formulated the need to discard a conflict. Some conflicts may be irrelevant for all involved stakeholders and, thus, they want to remove these conflicts from the system, without actually resolving the conflict. Once removed, this conflict should not be raised on subsequent model synchronizations.

Reflection: After this meeting, we adapted the states of a task adding a new state ‘ignored’ to denote that despite its (generated) due date, this task is irrelevant for the stakeholders, i.e. they agree to ignore the conflict (cf. Figure 5.4 on p. 140). Although stakeholders mutually agree to ignore conflicts that refer to a MODELELEMENT, they still pose inconsistencies. Thus, we also modified states of a MODELELEMENT to denote elements which are not involved in conflicts that should immediately be resolved, but, however, are involved in ignored conflicts. This intermediate state is denoted ‘With ignored tasks only’ (cf. [Ki14, p. 27]). Our final adaptation to address this concern has been made in the visualizations. Since ignored conflicts could influence information quality, ignored tasks are still visualized. We use a different color to denote that these tasks do not have to be addressed immediately (cf. Figure 7.5(b)). The color has been chosen in analogy to traffic-lights, i.e. yellow tasks are less critical than red ones (cf. Figure 7.5(a)).

7.2.3.3 Strategies

Conflict resolution strategies of MODELGLUE:

Context: The matrix to configure the conflict resolution strategy (cf. Figure 5.28 on p. 194) is a central element of the UI and meant to provide means for adapting and configuring how the system resolves conflicts. The matrix not only serves an adaptation of the conflict resolution strategy but also displays the behavior of the predefined strategies.

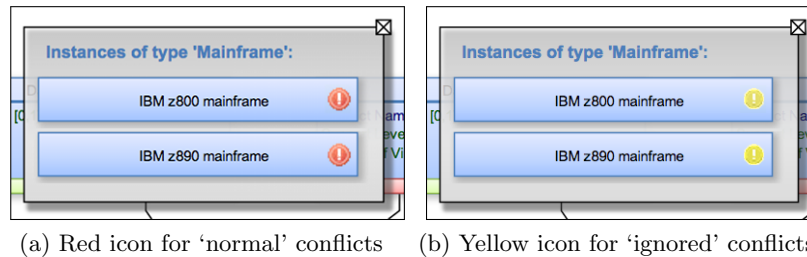


Figure 7.5: Visual representation of MODELELEMENTS with conflicts [Ki14, p. 81]

Feedback: “Despite its size and complexity the conflict resolution strategy matrix was intuitively understood [...]” [Ki14]. Both enterprise architects agreed that an adaptation of the strategy which determines how to resolve a conflict is important. They appreciated the predefined strategies and in particular regarded the characteristics of the tolerant strategy meaningful.

Reflection: It is beyond the scope of the present thesis to find an appropriate conflict resolution strategy for organizations or specific industry sectors. Thus, not each cell of the matrix has been discussed in depth. The characteristics of both strategies, i.e. tolerant and strict, have been pointed out. Further research has to show the adaption-behavior of an organization with respect to identified conflict (resolution) patterns (cf. Section 8.3).

Dialogs for adapting conflict resolution strategies in MODELGLUE:

Context: In Section 5.2.8.4.2 we described our design for a UI to (re)define rules that serve to merge models in the course of conflicting changes. As an alternative, we provide direct access to the underlying MxL queries (cf. [Re13] for the specification of MxL). Figure 7.10 on p. 278 depicts two such MxL queries. On the one hand, we wanted to understand whether our UI design is intuitive, on the other hand, we intended to test whether the expressive power is sufficient for EA management tasks.

Feedback: An enterprise architect pointed out that one has to distinguish between power users and other users like data owners, who only occasionally work with the EA repository. While enterprise architects certainly are willing to learn new language concepts rather occasional users, however, have little interest in learning a ‘cryptic’ language for ‘yet another tool’. The enterprise architect confirmed that for ‘simple’ EA models and basic queries the dialog is adequate. In more complex scenarios, direct access to MxL may be useful.

Reflection: The UI design has been confirmed. MxL was designed without having write access in mind. However, this has been prototypically implemented for MODELGLUE and could be further extended in the future.

Due-date of conflicts in MODELGLUE:

Context: The life-cycle of a conflict was discussed with regard to responsiveness and reluctance of data owners and EA stakeholders. The enterprise architects raised the question how the system reacts if conflicts get never solved.

Feedback: Ideally, conflicts immediately get resolved to work towards CCMC. However, the enterprise architects provided us with examples aggravating conflict resolution; these range from stakeholder reluctance to complex changes on information systems which require considerable efforts. Thus, “consolidation might even take years, due to few, but substantial changes of the architecture” [Ki14].

Reflection: After this particular interview, we equipped conflict tasks with a ‘due date’ and refined the states of tasks. We added the new state ‘overdue’ (Figure 5.4 on p. 140) which allows to determine unresolved tasks that are past their due date. Once overdue, it is possible to send reminders to the owner of the task or even escalate the task through their superior. The task might get re-assigned as it is delegated by the superior.

Quality metrics in MODELGLUE:

Context: We discussed possibilities to monitor the quality of the imported information.

Feedback: Ignoring conflicts could diminish model quality and in the long-run not only be counter-productive for EA management but pose a threat to the entire EA initiative. One of the enterprise architects proposed to introduce quality metrics.

Reflection: In his master’s thesis, Kirschner proposes three solutions, all involving the introduction of metrics over MODELELEMENTS and their conflicts. Since we regard the introduction of metrics beyond the scope of the present thesis, we refer to [Ki14, p. 81] for initial ideas on how to measure model quality including absolute, relative, and weighted (with respect to different conflict types) measures.

7.2.3.4 Conflict Resolution

Recurring conflicts in MODELGLUE:

Context: Related to the topics that centered around ignoring conflicts and long-lasting conflicts, we discussed how the system reacts in the course of consecutive synchronizations.

Feedback: The enterprise architects raised two major concerns: MODELGLUE might discard unresolved conflicts on subsequent synchronizations, or, alternatively, the exact same conflicts as in the previous synchronizations will be triggered again. This might lead to duplicate conflicts referring to the same issue.

Reflection: Kirschner’s notion of CIC helps to identify recurring conflicts. Since this information is stored in a task, MODELGLUE is able to analyze existing tasks before creating new ones (cf. Section 5.2.5.4).

Learning and batch-solving in MODELGLUE:

Context: Next, we discussed the support to resolve conflicts. In particular, we focused on Schrade’s learning mechanism (cf. Section 5.2.7.1).

Feedback: The enterprise architects endorsed the idea to support the resolution of many conflicts that may be resolved applying a very similar strategy. However, they also raised concerns about the transparency of the approach taken. They highlighted that the outcome is not clear and transparent for the user.

Reflection: On the one hand, users always apply changes to the tentative merge result in the preview model such that the changes can be viewed afterwards and ‘nothing can go wrong’. On the other hand, providing the user with a list that shows a preview of all actions to be performed *before* applying Schrade’s resolution strategy could improve MODELGLUE. This way, users could make adjustments, e.g. generally apply a strategy but remove particular exceptions. Kirschner [Ki14, p.82] also discusses the role of further heuristics and even suggests to provide facilities to define custom rules for the specification of heuristics. In his discussion, he also details an example. From a conceptual viewpoint, this would be a case in which the conflict resolution strategy has to be adapted to reflect more permanent resolution strategies. The heuristics could be a good indicator to identify patterns that help to further increase the degree of automation during conflict resolution. Similarly, a post mortem analysis of multiple conflict resolution sessions could give additional insights into frequently used resolution patterns of an organization.

7.2.3.5 User Interface Support

Next to the general understanding of MODELGLUE, we deep-dived into the visual support since we realized advanced UIs for differencing and conflict management. Again, Kirschner reports on this topic extensively in [Ki14, pp. 83–86] and we reflect his discussion from our perspective.

Visualization layers of the differencing visualization in MODELGLUE:

Context: The visual concepts and their behavior within the different interactive visualizations were discussed.

Feedback: The enterprise architects agreed to the layering concept as a means to realize a semantic zoom. The first layer was referred to as an intuitive way to browse through an EA model. They regarded it as a meaningful representation of the model when analyzing two different models. However, the enterprise architects also stated that the metamodels of information sources rarely change. From their perspective, the second layer is an important step to cope with the complexity of EA models. The third layer raised particular interest as the neighborhood of an OBJECT is highly important. The enterprise architects even talked about visual impact analyses at this layer. In line with Section 2.1.1, an enterprise architect confirmed that changes to attributes and values are not that

important for EA management. At the same time, these are the changes that occur most frequently. Against this background, a visual representation of these changes is generally a good idea. One particular suggestion to our design was made by the other enterprise architect. The background-colors of the three-way comparison are rather confusing. The background colors imply semantics (see Figure 7.6(a)) as some of these colors are used to refer to objects which have been changed (orange), created (green), or deleted (red).

Reflection: Generally, our design goals have been met. However, to our surprise the coloring of the three-way differences raised concerns. While our original intent was to use the background color to underpin that these are different versions of the same OBJECT, the colors seem to confuse the user. Since the operation that has been performed on an OBJECT is expressed by the visual element's color itself, we adapted our design to a more decent version depicted in Figure 7.6(b).

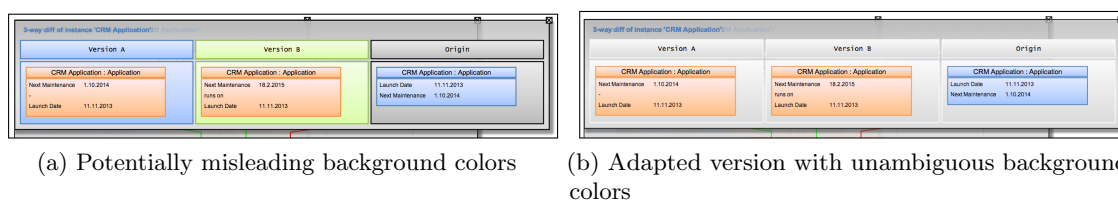


Figure 7.6: Adaptations of the differencing visualization in the three-way comparison [Ki14, p. 84]

As we pursue a uniform design for both, the differencing visualization as well as the conflict management dashboard, the feedback for differencing visualization also applies for the conflict management dashboard.

Layout and order of metamodel elements in MODELGLUE:

Context: The exact layout of the first layer of the conflict management dashboard was discussed with regard to the placement of visual elements representing OBJECTDEFINITIONS.

Feedback: The enterprise architects noted that the layout could be perceived by end users as being randomly positioned. According to one enterprise architect, users expect elements of a low abstraction level, e.g. the technical infrastructure, should be placed at the bottom while elements of a higher level, e.g. business processes and capabilities, should appear at the top. This perception is similar to the arrangement illustrated in Figure 2.2 on p. 17. Although power users of MODELGLUE, like enterprise architects, would cope with the pseudo-random allocation, occasional users, would profit from a layout that adheres to layers with strict (implied) semantics. Though the ideal layout may depend on the specific modeling community. A data owner of a CMDB may have a different viewpoint than one of an ESB.

Reflection: An alternative layout could assign the different OBJECTDEFINITIONS to a layer or even a cell in a matrix. A major drawback of such a solution is that line crossings between visualized relationships can only be optimized with respect to the defined arrangements, e.g. the corner stitching algorithm [Ou84] could be used to enhance the visual appearance of these lines. This algorithm is known from very-large-scale integration (VLSI) design and seeks to create an integrated circuit by combining thousands of transistors into a single chip meanwhile minimizing wire, i.e. line, crossings. However, from our experience with system cartography [Ma08, MBL⁺08, RZM14] and against the background of our own development efforts in this research direction [SMR12, HMR⁺12, HRP⁺13a, RHZ⁺13] we draw the conclusion that none of these solutions produces an automatic layout, which conforms to predefined layers or cells and at the same time is visual appealing. The underlying technical issues can be regarded complex and there is a considerable amount of research endeavors investigating visual appearance and layouts in graph drawing [PMC⁺01]. Since the metamodel is not modified all too often, one could combine layout algorithms with manual adaptations, i.e. initial algorithmic layouts with ex-post user adaptations that are reapplied until the underlying metamodel changes.

Visualizing constraint violations in MODELGLUE:

Context: Different ways to ensure model quality and consistency were discussed. In particular, we proposed to illustrate constraint violations in the merge result visualizations.

Feedback: One enterprise architect argued that constraint violations represent a different viewpoint; such a viewpoint was beyond the scope of their EA management initiative. In the enterprise architect's organization, a separate role was responsible for quality aspects, documentation, and other formalities. This role was accountable for constraint violations and their management. The enterprise architect further continued that this role would profit from similar visualizations that only focus on constraint violations. The EA team was not interested in such information and was rather concentrated on model conflicts.

Reflection: Since MODELGLUE already is aware of constraint violations, it is certainly possible to visualize these violations. This would include minor adaptations in the DATABINDING of the difference visualization as well as the respective ABSTRACTVIEWPOINT (cf. Section 6.3). To our surprise, it seems that it is beyond the scope of this EA management initiative to cope with constraint violations. However, further research might prove utility for a different role within this organization.

Allowed actions to resolve conflicts in MODELGLUE:

Context: The different actions of a model conflict were discussed.

Feedback: In addition to provided actions, one architect motivated additional functionality to resolve conflicts. A person in charge of the conflict resolution could want to perform arbitrary changes to resolve a conflict. Additional changes could

be considered more appropriate than the choices offered, e.g. a mixture of two values for a single attribute.

Reflection: Our current design addresses such a scenario. We provide links to the MODELELEMENTS such that arbitrary changes can be applied. The user must then manually set the task to a different state.

Tabular view vs. interactive conflict management dashboard of MODELGLUE:

Context: Although MODELGLUE’s UIs to resolve conflicts follow a coherent design, the UIs follow two fundamentally different paradigms. One is realized as an interactive visualization with layered graph and tabular layouts (cf. Section 5.3.2), whereas an alternative view realizes a tabular conflict list (cf. Figure 6.9 on p. 223) along with a link to details of each conflict (cf. Figure 6.10 on p. 224).

Feedback: One enterprise architect considers the interactive visualization as an adequate means to get an overview of the model conflicts and refers to it as the ideal tool for a manager whose intent is to identify problems in the EA model. However, conflict tasks most likely will not be solved by a managerial role. The enterprise architect further explains that for such a task only little knowledge is required and the assignee of a conflict resolution task perhaps does not have access-rights to the entire EA model. In such a situation—particularly if many tasks must be resolved—the tabular view would be prevalent in the analyzed organization.

Reflection: During the design, it was our intent to incorporate little information in the interactive visualization. However, the amount of information should be sufficient for deciding on a conflict. The details view (cf. Figure 6.10 on p. 224) can be reached from within the interactive visualization.

7.2.3.6 Decontextualization

EA-specific model modifications in MODELGLUE:

Context: The rule-based approach to detect conflicts was the starting point for a discussion that centered around the application of similar rules for general EA modeling.

Feedback: Both enterprise architects deemed the application of rules to the EA model as important. It would provide utility to the EA team in particular to adapt many MODELELEMENTS at once. Kirschner reports an example: “to add an attribute ‘application life-cycle’ with the possible values ‘plan’, ‘preparation’, ‘implementation’ and ‘operation’ for every element of type ‘application’” [Ki14, pp. 82,83]. He further states that these rules also could be applied to transform different formats or serve to perform ‘one time’ lookups.

Reflection: Respective adaptations that would have been necessary to implement such a rule-based model-to-model transformation center around the UI of the dialogs presented in Section 5.2.8.4.2. Although we estimated the efforts of these adaptations to be very little, we regarded them as beyond the scope of

our research and more in scope of a common EA repository. Thus, we did not implement required changes in our prototype.

Decontextualizing MODELGLUE:

Context: It was discussed whether MODELGLUE could be applied in the context of other modeling communities to support their planning habits with a model-driven tool.

Feedback: Both Enterprise Architects agreed with the general idea but also made us aware of possible reluctance of stakeholders since MODELGLUE could be perceived as ‘yet another tool’. Hence, they were in doubt that anyone except the EA team would apply MODELGLUE.

Reflection: On the one hand, we stated in Assumption 4.4, that we consider EA management is supported by upper management and, thus, the EA team has considerable impact on the modeling communities. Hence, working with MODELGLUE could be enforced by additional incentives. On the other hand, using MODELGLUE in a different context than EA model maintenance goes beyond the scope of the present thesis (cf. Section 8.3).

7.2.4 Reflection of Results

Reflecting on the feedback of the case study outlined above, from our perspective, only minor adaptations were required, i.e. the major design decisions seem to be promising with respect to this specific use case. In [Ki14, p. 76], Kirschner additionally sketches a complexity analysis of the data as well as the trends concerning number of instances. He confirms the order of magnitudes provide in [RM14] for the different visualization layers (cf. also Section 5.3.1 and Section 5.3.2).

Although the case study took place in a highly specific environment as well as the feedback is highly subjective, we gained some insights in minor design flaws which we deem plausible and are rather obvious retrospectively viewed, e.g. the background color of the three-way differencing (cf. Section 7.2.3.5). Also, we identified parts of our design which can be applied in different contexts, e.g. the definitions of model-to-model transformation rules using the UI as illustrated in Section 7.2.3.6.

7.3 Case Study in the German Health Industry

The second case study took place in an organization of the German health industry. Neubert [Ne12, p. 167ff] reports on the EA management initiative of this organization in the context of emergent modeling. While Kirschner reports on the case study extensively, we briefly summarize his findings and discuss important points that influenced our design and provide additional insights. For detailed information on the organization and setup of the case study, we refer to Kirschner [Ki14, pp. 87–98].

7.3.1 Initial Interviews

In an initial one-hour phone call, we outlined the concepts behind Federated EA Model Management and MODELGLUE. We introduced MODELGLUE to a senior enterprise architect of the organization. Since we already were familiar with the environment of the organization, we envisioned a good fit between our conceptual work, MODELGLUE and the EA initiative of the organization. The selection of subjects certainly influences the results of case studies (cf. Section 7.4 and [RH09]). However, our goal to gather expert feedback could still be met. The limitations and validity and general application of our findings are discussed in Section 7.4.

During the second part, we mapped the proposed concept to the environment of the organization and outlined specific use cases it should support. That left us with a precise scope for the case study and application of MODELGLUE. Similar to the first case study reported in Section 7.2, a LOI has been exchanged followed by a more formal NDA, since the models sent to us by the enterprise architect may contained confidential information. For the exchange of models, the organization relied on CSV and XLS/XLSX files.

7.3.2 Context

Before we summarize our findings in Section 7.3.3, we describe the environment as well as the EA model of the organization. The context provides important insights in an application scenario for Federated EA Model Management.

7.3.2.1 Federated Modeling Environment of the Organization

The organization pursues EA management for several years. They employ an EA repository that is metamodel-based and capable to model similar flexible structures than MODELGLUE³.

The organization's federated EA model environment consists of an SAP FI system that is currently synchronized with the EA repository, i.e. information of the SAP FI system is part of an entity stored within the EA repository. In particular they employ this entity to calculate forecasts on made assumptions and trend analyses of the SAP FI system. This forecasting is done using the EA repository whereas the actual costs are stored in SAP FI. As we found out, information in SAP FI as well as in the EA repository could be changed concurrently.

Besides SAP FI, the EA repository is synchronized with a CMDB. This synchronization takes place in a unidirectional manner whereas the imported information is considered read-only, i.e. the EA repository changes are discarded if deviations with the CMDB occur during the next synchronization.

Additional repositories are integrated with the EA repository: The enterprise architect currently maintains ten models within the EA repository, with approximately 50 OBJECT-DEFINITIONS. For this case study, we decided to limit the scope to these two systems and the respective process support.

³More precisely, the predecessor of MODELGLUE is applied in this organization, cf. also [Ne12, p. 167ff].

7.3.2.2 Examined Part of the EA Model

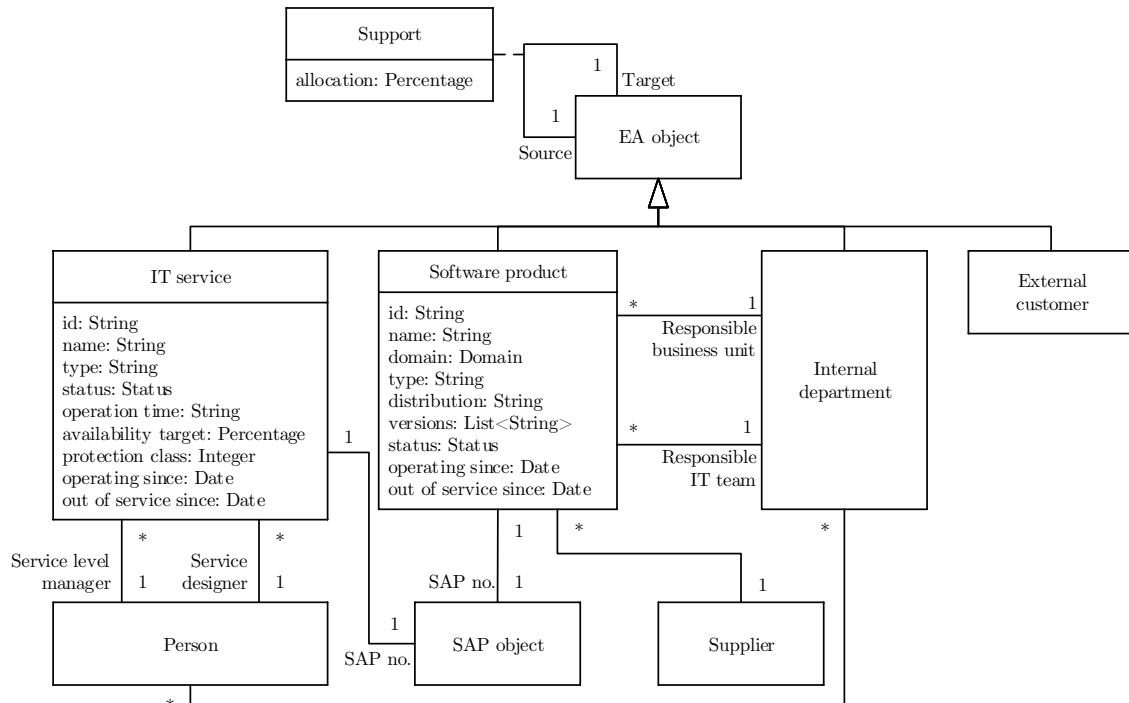


Figure 7.7: Excerpt of the EA metamodel of the organization [Ki14, p. 95]

The enterprise architect provided us with a database dump representing an excerpt of their current EA model for further analysis. As depicted in Figure 7.7, key classes of the metamodel are SOFTWARE PRODUCTS and IT SERVICES. Additionally, the association class SUPPORT is of high importance to describe the relationship between the different EA OBJECTS, which is the common class for all EA model elements. The SUPPORT class captures the allocation level between two EA OBJECTS. Thereby, INTERNAL DEPARTMENTS are referenced if the costs have been produced in the respective unit of the organization or alternatively an EXTERNAL CUSTOMER is referenced if the costs have been produced by an entity outside the organization.

In his master’s thesis, Kirschner provides an additional analysis of the given models [Ki14, p. 95ff] with respect to their structure, number of instances as well as performance considerations. His analysis reveals that our estimated orders of magnitude for each layer as proposed in [RM14], Section 5.3.1, and Section 5.3.2 prevail. During his analysis, Kirschner provides concrete examples as well as additional rationales for our visual designs. We want to emphasize one particular design decision based on one of Kirschner’s illustrations which is depicted in Figure 7.8. It shows the second layer, i.e. the list of all objects—regardless of whether model conflicts or differences are viewed. As we stated during the design, we eliminated every unnecessary detail, except color or an icon indicating its state to other objects or conflicts. However, during the design of the visualizations, we considered additional mechanisms to filter objects a necessity although a lean layout has been chosen (cf. Section 5.3.1.5 and 5.3.2.6).

7. Evaluation

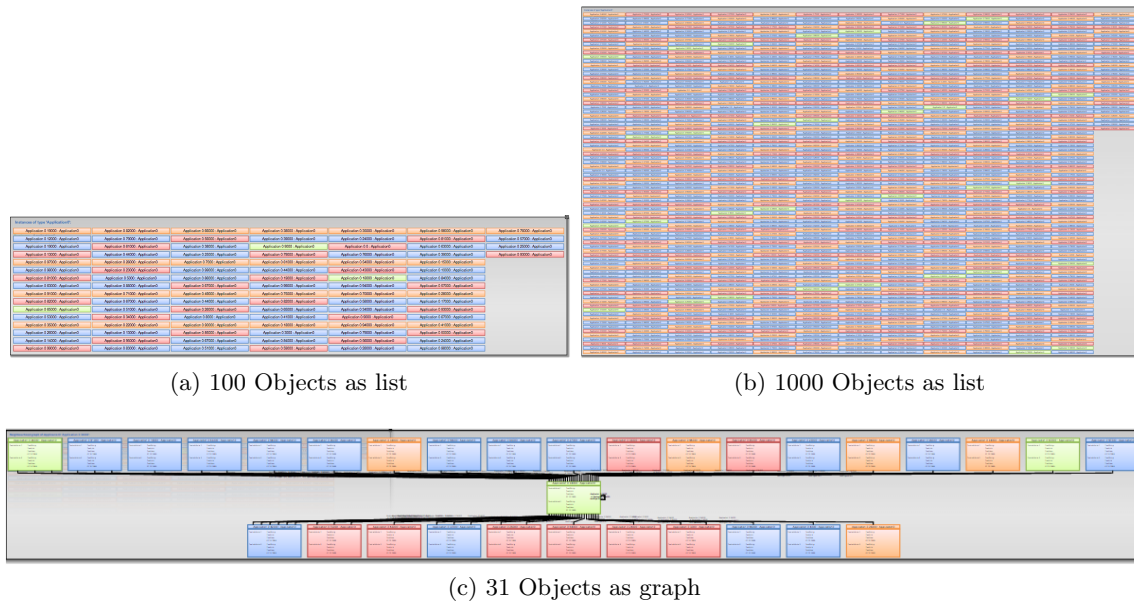


Figure 7.8: Different quantities of objects in the second and third layer of the visualizations [Ki14, pp. 78,98]

Compared to the graph that also shows the neighborhood as well as the attributes of an object (cf. Figure 7.8(c)) the list performs well at an order of magnitude of 10^2 (cf. Figure 7.8(a)). However, as the list grows considerably large $> 10^3$ Objects, information overload (see p. 126) cannot be prevented (cf. Figure 7.8(b)). To cope with more instances at the second layer, we introduced the filter dialog. As stated correctly by Kirschner, any graph layout cannot be applied at the second layer.

7.3.3 Execution, Feedback, and Adaptations

Federated EA Model Management process that guides MODELGLUE:

Context: The process behind Federated EA Model Management (cf. Section 5.2) was discussed.

Feedback: In the organization, the EA team triggers the manual delivery of a dump from an information source. Provision of information is based on a mutual agreement including selection of the intended part of an information source's model that serves to address the concerns of EA stakeholders. Technical questions, i.e. exact mapping, format, etc. were dealt with informally. Information is gathered on a monthly or quarterly basis. Export mechanisms within the information source select and possibly aggregate information. The produced result is delivered in CSV format. The enterprise architect also revealed that currently a spreadsheet application with customized scripts is employed to ensure model quality, i.e. view differences to previous versions. Moreover, the enterprise

architect noted that obvious problems can be detected with the differencing visualization. If inspected successfully, the information is pushed into the EA repository. In contrast, if this quality check fails, the dump is discarded, the error identified, and a new dump generated.

Reflection: The description of the rather implicit process, the enterprise architect's organization follows to maintain the EA model, confirms our process design detailed in Section 5.2. Moreover, the enterprise architect assigned specific software artifacts we provided, to the concrete activities. We found that the integration of information sources including the logical and physical mapping is done in an informal manner. Given the size of the organization and the size of the (analyzed) EA model, pursuing an ad hoc approach could be beneficial.

Incremental extension of the scope of an EA model:

Context: Our process design is not only iteratively but also advises to integrate information sources gradually and thereby extend the EA model incrementally.

Feedback: The enterprise architect agrees to limit the scope of an EA model initially. Details are far less important than a quick overview of the entire EA. However, cumulated details in turn could provide considerable insights in an EA, too (cf. Section 2.1.1).

Reflection: This shows that an organization pursuing Federated EA Model Management should choose information sources by considering the (prioritized) concerns of EA stakeholders to provide benefits and show early results (cf. [HRS⁺14, HMR12]).

Integrating information sources in MODELGLUE:

Context: The integration of information sources was extensively discussed.

Feedback: The enterprise architect advocates a lean approach. Loose coupled interfaces via dumps are less error-prone than hard-wired interfaces, e.g. Representational State Transfer (REST) interfaces. The organization currently applies both, database dumps in the form of file exports as well as fully integrated (uni-directional) interfaces that push information to the EA repository. Having experienced both, the enterprise architect favors the exchange of files, e.g. XLS/XLSX or CSV files, that contain the desired information.

Reflection: MODELGLUE has a variety of import mechanisms. Although the present thesis did not reveal every detail, considerable experiences with import from XLS/XLSX or relational databases have been made, e.g. [Or13]. For this case study, we adapted existing import mechanisms to import CSV files which have been provided to us by the enterprise architect.

Information flow in a federation:

Context: In Section 5.2.2.3, we stated two fundamental ways to exchange information: unidirectional and bidirectional.

Feedback: According to the enterprise architect, only unidirectional flow of information is feasible, i.e. from information sources to the EA repository. Automated change propagation in the opposite direction would be far too complex as well as implementing bidirectional interfaces would perhaps be more fragile and raise additional questions, e.g. security, ownership, impact of modifications on productive information etc.

Reflection: In Assumption 4.3 we also state that—from a technical perspective—we advocate a unidirectional synchronization. We foresee that the integration of new information sources is a major cost driver for Federated EA Model Management. However, propagation of (consolidated) changes should logically flow in both directions. Thus, we agree with the enterprise architect.

Intermediate models vs. links to other information systems in MODELGLUE:

Context: In Section 4.2.5, we outlined the difference between using an ontology to reference content in third party systems on the one hand and copying content to an import model such that information is stored in the federal system on the other hand.

Feedback: The enterprise architect stated that intermediate models are required and direct mappings are not feasible with respect to the architect's organizational context.

Reflection: Since we already discussed both alternatives and their disadvantages (cf. Section 4.2.5), we appreciate this statement as it is confirming our design.

Granularity of access-rights in MODELGLUE:

Context: We discussed the different roles and respective access rights with the enterprise architect.

Feedback: The enterprise architect pointed out that a fine-grained access control is of utmost importance. Almost always, the enterprise architect, underlined his opinion, giving access-rights just on the granularity of OBJECTDEFINITIONS is wrong. This also holds true for access-rights on OBJECTS as these often contain ATTRIBUTES that should not be visible for all EA stakeholders and would be either irritating or would raise additional concerns, e.g. financial goals or performance measures.

Reflection: The enterprise architect confirms our design decisions since we consider ATTRIBUTES to be the finest granularity for access-rights. During our design, we also considered access-rights on values. However, throughout the case studies and the interview series it turned out that practitioners do not demand such access-rights on values.

7.3.3.1 Tasks

Task handling in MODELGLUE:

Context: The role of the tasks during conflict resolution was discussed.

Feedback: The enterprise architect noted that the EA repository was used by the EA team primarily and other business units within the organization showed a rather selective usage behavior. Further, the enterprise architect noted that the person who initiates the merge process also resolves arising conflicts. If that resolution process involves other parties, the architect directly contacts the respective modeling community via phone or e-mail.

Reflection: In Assumption 4.4 we stated that the EA initiative has upper management support; further, in Assumption 1.1 and Assumption 4.7 we detail that we assume all communities support EA management by providing information as well as by working towards an envisioned state of CCMC. Collaborative model merging (including task assignment) of MODELGLUE depends on the support by the modeling communities and, thus, the organizational culture. MODELGLUE must be established and accepted as a means to improve the decision base for EA management, i.e. the EA model. During the discussion, we concluded that the tasks are ‘a nice idea’, but within this specific organization, e-mail is the de facto collaboration tool. After this feedback, we equipped the tasks with an e-mail notification containing a direct link to the respective task in MODELGLUE.

Conflict task life-cycle in MODELGLUE:

Context: In the case study reported in Section 7.2, we found the need for long-living conflicts or even for ignoring conflicts within MODELGLUE.

Feedback: The enterprise architect noted that permanently ignoring conflicts is not an option in his organization. One must ensure the quality of the EA model, i.e. conflicts may easily exist for several weeks, but in the end they are resolved.

Reflection: The EA expert’s feedback showed us the wide-range of opinions since the interviewee states the exact opposite to the first case study. Since the introduced state ‘ignore’ does not affect operations of the remaining system (assumed no conflict is ignored), we kept the feature in. More general, contrary meanings and opinions indicate an additional configuration point for MODELGLUE and Federated EA Model Management.

Tabular view of conflicts in MODELGLUE:

Context: The enterprise architect suggested a specific design to deal with conflicts not knowing the alternative tabular view of MODELGLUE.

Feedback: The enterprise architect favored a tabular view of conflicts (cf. Figure 6.9 on p. 223).

Reflection: We implemented both views right from the start of our development efforts and envisioned users that prefer tabular views over more sophisticated visualizations. The enterprise architect confirmed our design to provide both, interactive visualizations as well as tabular views on model conflicts.

7.3.3.2 Strategies

Conflict detection in MODELGLUE:

Context: The conflict detection in the course of a merge process was discussed.

Feedback: The enterprise architect was rather skeptical concerning an automated detection of model conflicts. If the order of magnitude of conflicts is below 10^2 , the enterprise architect argues, either a conceptual mistake or technical error slipped in the process. In this vein, none of the conflicts are actually solved but the root cause is eliminated such that subsequent synchronizations succeed.

Reflection: MODELGLUE merges information in a model containing tentative merge results. This model could be discarded if severe problems are diagnosed. As stated in Section 5.2.5, involved models are not altered until the merge process is finished.

Preferred model during a merge in MODELGLUE:

Context: The conflict resolution strategies were discussed.

Feedback: The enterprise architect emphasized the importance to specify a ‘preferred model’ whose changes supersede changes of all other models when merging them. This could lead to a substantial decrease of conflicts. The interviewee also argued for a clear separation of responsibilities, i.e. to use one primary information source for OBJECTS of a particular OBJECTDEFINITION only. The architect estimates that 90% of imported changes can be applied from a ‘primary information source’.

Reflection: One way to permanently define such a ‘preferred model’ is to adapt the conflict resolution strategy; this is a rather long-lasting solution which was not the intent of the enterprise architect. Instead, the learning mechanism could help to reduce the manual effort. However, during the discussions, it turned out that this solution is also not quite well designed for this purpose. Thus, we introduced a way to specify a ‘preferred model’ whose changes are privileged throughout the merge (cf. Figure 6.8 on p. 222). This preference supersedes any configured conflict resolution strategy. The concept mentioned by the enterprise architect was coined *master-slave* and could be confirmed as prevalent solution for conflicts in our additional interview series (see Section 7.5).

7.3.3.3 Conflict Resolution

The interviewee stated that for the general resolution of conflicts between information sources and EA repository, the enterprise architect is in charge. Phone or e-mail is the common means for communication in their organization.

Boundary model elements in MODELGLUE:

Context: We further discussed the aspect of overlapping model elements and responsibilities, i.e. two or more information sources for one real-world object.

Feedback: The enterprise architect stated that data owners of information sources involved in conflicts are consulted directly. An often chosen solution is to establish a master-slave relationship between two or more information sources. Necessary changes on an information source are performed by the data owner.

Reflection: MODELGLUE does not explicitly support master-slave paradigms (see above). However, a customized merge rule (cf. Section 5.2.8.4.2) could be used to implement such a scenario.

7.3.3.4 User Interface Support

Similar to the first case study, the UI designs have been discussed in detail. Thereby, we put focus on the interactive visualizations.

Legend for visual concepts in MODELGLUE:

Context: The visualizations were discussed with a particular focus on the semantics of different symbols and colors.

Feedback: The enterprise architect noted that the semantics could be clarified by adding a legend to the visualization.

Reflection: Adding a legend would foster initial comprehension; however it would limit the available size on the screen to display information. Since we strove for an intuitive design, we did not include a legend. We conclude that, especially for novel users one could add a legend that could also be turned off for power users.

Transparency of visual overlays in MODELGLUE:

Context: The different layers of the visualization were discussed in detail.

Feedback: The high transparency of overlay windows influences visual perception negatively. Especially crossing lines with layers shown below the layer currently active is rather confusing.

Reflection: We altered our initial design depicted in Figure 7.9(a) to an adapted version illustrated in Figure 7.9(b). The design shown in Figure 7.9(b) incorporates not only solid colors, but uses a drop shadow to emphasize separation between layers. This was part of a broader visual re-design of the solution such that many minor adaptations lead to a more decent appearance.

7. Evaluation

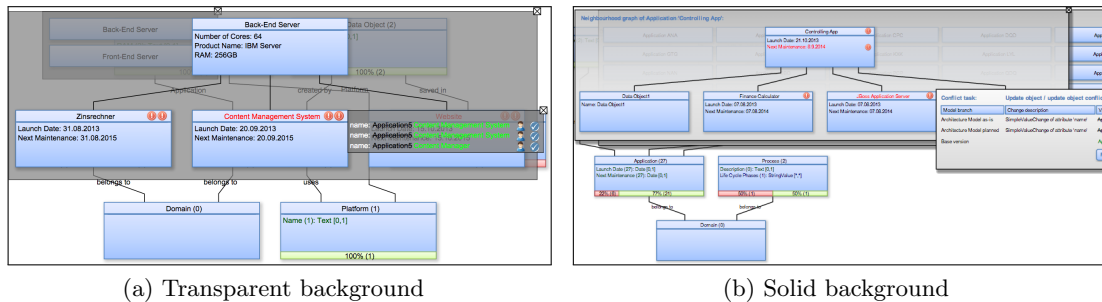


Figure 7.9: Background colors of the overlay windows within the visualizations [Ki14, pp. 93]

Filter in MODELGLUE:

Context: In Section 5.3.1.5 we proposed a design to filter OBJECTS based on user-defined rules. We strove for a self-explanatory and intuitive design meanwhile offering the required expressive power for EA models.

Feedback: The enterprise architect confirms that the UI is intuitive and any enterprise architect should understand how to use it without further explanations. Concerning the expressive power of the filter, the enterprise architect could not think of counter-examples and regarded it as sufficient for its purpose.

Reflection: This confirms our design; from our perspective, the expressive power of the filter needs to be studied in a broader field.

Filtering by constraint violations in MODELGLUE:

Context: In the course of the discussion about the filter, the enterprise architect suggested some improvements.

Feedback: In order to improve the quality of the EA model, one could filter by CONSTRAINTVIOLATIONS (cf. Section 5.3.2.6). The interviewee also recommended to include more details concerning these violations, e.g. filter by objects that miss mandatory attributes. A respective visualization could provide additional insights in the quality of the model.

Reflection: The suggestions of the enterprise architect follow our vision to assess the quality of models visually. A concrete realization of this idea would require the development of a new ABSTRACTVIEWPOINT (cf. Section 6.3). Assuming that this visualization type would look similar to the differencing visualization, a new DATABINDING, which is a minor adaptation of the existing DATABINDING, would be required.

Definition of global filters in MODELGLUE:

Context: Our design illustrated in Figure 5.3.1.5 on p. 204 presumes that one intends to query for OBJECTS that conform to a specific OBJECTDEFINITION.

Feedback: The enterprise architect formulated the demands for a filter mechanism that queries across OBJECTDEFINITIONS. A concrete example was provided to us: a query which shows all OBJECTS containing ‘SAP’ in their name regardless of the respective OBJECTDEFINITION.

Reflection: The enterprise architect identified a potential limitation of the filter UI or an extension thereof. In the example outlined above, with the current design, a user must define rules that filter every OBJECTDEFINITION. Cross-OBJECTDEFINITION filters are currently not supported by MODELGLUE. However, minor changes on the algorithms as well as the UI could realize such a feature.

Save filter queries in MODELGLUE:

Context: Brainstorming about ideas how to further improve the filter, the demand to persist (filter) queries came up.

Feedback: The enterprise architect thought it would be beneficial if the filter is stored within the system and could be accessed quickly. It is very likely that users apply the exact same filter multiple times; persisting the filter (or the configuration thereof) would accelerate their workflow.

Reflection: Filters are represented as JSON strings (cf. Listing 5.1 on p. 206 and [RM14]). The UI is already programmed to reload the entire configuration including existing filter parameters. Thus, the configuration of the dialog, i.e. the JSON query, could be persisted, loaded, and easily reapplied.

MxL expressions for advanced filtering in MODELGLUE:

Context: MxL provides access to define queries that are beyond the expressive power of the presented UIs (cf. Section 5.2.8.4.2). We provided two examples illustrated in Figure 7.10 to the enterprise architect. Thereby, the query shown in Figure 7.10(a) can be configured with the UI whereas the query depicted in Figure 7.10(b) includes a condition on an ATTRIBUTE only available by traversing transitive relationships. Note that the expressive power of the latter query is beyond the configuration capabilities of the proposed UI.

Feedback: The enterprise architect referred to MxL as not particularly user-friendly and named reasons why MxL would provide little benefit in the organization: First, very few people were engaged in EAM. As a consequence, the EA repository had few power users. Second, the motivation of occasional users to learn a sophisticated language was rather low when the language could only be applied to define queries in the already rarely used EA repository. Third, MxL was not widespread, which would make finding and resolving syntactical issues rather difficult and posed an additional threat. The enterprise architect envisioned an active community answering MxL related problems.

Reflection: We employ MxL in dialogs as a means to an end. Thereby, intermediate templates are used to define queries in MxL syntax. Thus—from an architectural point of view—designers can vary the amount of MxL that is accessible or shown

to the user. Given the feedback of the EA expert, one should invest in advanced UIs that build upon existing query languages.

```
find(Application)
  .where([("Launch Date" > "01.05.2014" and "Launch Date" < "30.05.2014") or
         ("Next Maintenance" > "01.05.2014" and "Next Maintenance" < "30.05.2014")])
```

(a) Range query

```
find (Server)
  .select([
    get Application whereis Deployed.any("Development cost" > 100000)])
```

(b) Transitive query

Figure 7.10: Exemplified MxL expressions [Ki14, pp. 66]

Conflict management dashboard of MODELGLUE:

Context: The visual elements of the conflict management dashboard were discussed. Particular focus was put on the progress bar beneath an OBJECTDEFINITION (cf. Figure 5.41 on p. 208). It is used as an indicator for conflicts on OBJECTS.

Feedback: The enterprise architect appreciates the idea and referred to it as a means to find ‘hot spots’ guiding user’s attention towards elements with the most conflicts in the EA model. “This fosters [...] model quality assessment and conflict identification” [Ki14, p. 94].

Reflection: This positive feedback confirms our design.

Two-way or three-way differencing in MODELGLUE:

Context: Both, two-way and three-way comparison is possible when calculating differences (cf. Section 2.3.1). In our prototypical implementation, we employ both techniques.

Feedback: The enterprise architect stated a two-way-comparison is sufficient in the context of EA models; the additional option (the display of two MODELELEMENTS’ origin) could be rather irritating for users.

Reflection: The underlying visualization stack of MODELGLUE is highly configurable (cf. also [RHZ⁺13]) such that users can configure to hide the display of the common origin of a MODELELEMENT. The framework has to be adapted with regards to user-specific preferences. Currently, the visualization framework applies persistent configurations globally and does not allow to define user-specific preferences.

7.3.3.5 Decontextualization

Decoupling the filter from the visualizations in MODELGLUE:

Context: It was discussed that filtering OBJECTS by specific values of their ATTRIBUTES is a common use case of all EA repositories.

Feedback: Demands for additional filtering capabilities of the EA repository were raised by the enterprise architect. The enterprise architect outlined that one does not always work inside a visualization. Demands for filtering the same information in different views were raised, i.e. defining queries with the filter UI for other views would provide utility for the enterprise architect.

Reflection: This might be an indicator of limitations of the EA repository currently used by the enterprise architect. In particular the platform does not allow to define queries combining multiple logical boolean expressions via conjunctions or disjunctions (**and/or**). Future research might decontextualize our UI design for mere filtering purposes. For further readings of filtering OBJECTS on the underlying platform in an efficient manner, we refer to Steinhoff [St13].

Deriving planned states with MODELGLUE:

Context: In Section 8.3 we point out that another application for MODELGLUE could be the modeling of planned states. Once applied or realized by a transformation project, a planned state could be merged with the current state of the EA model.

Feedback: The enterprise architect agreed with the general idea, especially having a branch of the metamodel could be beneficial. However, concerns were raised if the model must be branched, too. In the opinion of the enterprise architect, for mere planning purposes the model would not be required.

Reflection: MODELGLUE could serve the planning purposes of the enterprise architect. As a naïve approach, one could delete OBJECTS and their ATTRIBUTES after branching. Further considerations would suggest to decouple the branching of model and metamodel as proposed by Kirschner [Ki14, p. 90].

Towards a VDSL for meta modeling in MODELGLUE:

Context: Since the proposed visualizations provide considerable facilities to manipulate the underlying model (cf. Section 6.3), the discussion centered around altering the metamodel visually.

Feedback: The enterprise architect noted that MODELGLUE currently does not allow a manipulation of the metamodel via its visualizations. This could facilitate the creation of planned states of an EA. “As most EA tools offer visual model manipulation capabilities, users would expect such functionality [...], too.” [Ki14, p. 90].

Reflection: Extensive manipulation of the metamodel via visual concepts is beyond the scope of our research presented in this thesis. However, the developed framework (cf. Section 6.3) is designed to support visual interactions that propagate changes to its underlying model. One could realize a VDSL with this framework [SMR12, RHZ⁺13].

7.3.4 Reflection of Results

This case study particularly shaped how we perceive the process support for Federated EA Model Management and its interrelationship with MODELGLUE. Providing his expertise,

the senior enterprise architect gave us useful insights how the created artifacts can be integrated in this method in a meaningful way. These findings considerably shaped the process as well as the structure of Section 5.2 in the present thesis.

Many expectations and ideas of the enterprise architect center around features that have a productive character and go beyond the purpose of our prototype. Sometimes the enterprise architect compared MODELGLUE with a fully-fledged EA repository and COTS products of the EA domain. While this certainly was none of our intentions, it shows the maturity of the prototypical implementation. However, it also shows that a certain set of features must be achieved by an EA repository which forms a barrier for case studies in this area of research.

7.4 Conclusion of the Case Studies

In this section, we conclude both case studies. Since the respective conclusions for each case study already have been drawn in Section 7.2.4 and Section 7.3.4, we stick to a meta-conclusion outlining our learnings.

An interesting observation in the initial phases of the case studies has been that both organizations mostly rely on exact name matching and rather lean formats, e.g. CSV or XLS/XLSX. In the light of this observation, we regard the application of ontologies to resolve a mapping problem in an organization as even more challenging (cf. Section 4.2.5).

Both case studies shaped our design considerably. While the first case study contributed to our understanding of necessary and important additional states of tasks, the second case study served to better understand interrelationships of process steps with the software support, we provide. In line with Kirschner [Ki14, p. 98], we conclude that all participating EA experts appreciate MODELGLUE's functionality and the case studies show that EA experts familiar with MODELGLUE agree that it can provide *substantial utility* for an applying organization.

The results have to be interpreted carefully as they state opinions of individuals with a specific background and organizational environment. Given the empirical basis, they give us clues for limitations, further improvements and future research; our findings cannot be generalized. From our perspective, we identified relevant configuration points rather than ultimate changes to our design.

Potential biases are that we knew both organizations as well as their EA initiatives quite well. Knowing their initiative, we envisioned a good fit between our approach to Federated EA Model Management, MODELGLUE, and their organization. However, we considered the selected organizations and respective EA experts since they already pursue similar endeavors to accelerate their EA model maintenance endeavor.

7.5 Interview Series

Espinosa et al. [EAB10] state that interviews are “widely used in information systems research” and give prominent examples ([MRM⁺00, MMC⁺01, Or93, Or02]). In line with Espinosa et al., we agree that interviews are a good method to gain additional insights in EA management specifics.

In [AK14, ch. 4], Aleatrati Khosroshahi presents results of an interview series among 11 interview candidates. We summarize the key findings of his research. The interview guidelines can be found in [AK14, Appendix A], the list of participants with respect to industry sector, role, and size of the organization in terms of number of employees is illustrated in [AK14, p. 55–58]. Therein, also the approach of the interview series as well as the method we followed are described.

7.5.1 Status-quo in Industry

Aleatrati Khosroshahi [AK14, p. 59–61] investigates the current state of EA management initiatives in organizations. His first finding describes a common threat for EA management endeavors, i.e. EA stakeholders from business have not (yet) grasped the benefits of an EA initiative. Besides a lack of general acceptance among business-focused stakeholders, he diagnoses that most of the interviewed organizations have not established a holistic and coherent approach for Federated EA Model Management. However, organizations pursue considerable efforts that can be characterized as closely related to Federated EA Model Management, e.g. import of information sources’ models to an EA repository.

7.5.2 Benefits of Federated EA Model Management

Next, Aleatrati Khosroshahi [AK14, p. 61–64] investigates the perceived benefits of Federated EA Model Management. Benefits described concern for instance:

- **IT Controlling**, e.g. increased transparency with respect to IT and the identification and quantification of unnecessary overheads,
- **Trends and Forecasting**, e.g. determining required storage space and projecting future demands,
- **Controlling and Planning the EA**, e.g. reducing heterogeneity within an EA,
- **Regulatory Requirements**, e.g. responding to new information demands faster whereas the EA model helps to determine where to get the information,
- ...

While his findings certainly can be attributed to an up-to-date EA model, a direct impact of methods and techniques of Federated EA Model Management could not be diagnosed.

7.5.3 Additional Process Steps

During his interviews, Aleatrati Khosroshahi [AK14, p. 65–67] also discussed essential process steps and their dependencies and transitions as well as the iterative nature of the Federated EA Model Management process. An important step, we identified throughout the interviews is the alignment of terminology prior to an integration of an information source (cf. Section 5.2.2.2). In his master’s thesis, Aleatrati Khosroshahi [AK14, p. 66] details this sub-process and proposes a process model based on the interview results. Besides this process step, practitioners agreed to the core process steps (cf. Section 5.2); the interviews also confirm that the exact order of execution is rather flexible.

7.5.4 Degree of Automation

Next to the process, Aleatrati Khosroshahi [AK14, p. 69–70] elaborates practitioner feedback that gives insights in important parameters to consider when integrating information sources with an EA repository in an automated fashion. Important factors to consider are:

- number of ATTRIBUTES of an OBJECT and respective OBJECTDEFINITION,
- number of links of an OBJECT and respective relationships within their OBJECTDEFINITIONS, and
- potential overlap with other information sources, i.e. other models that describe the same real-world object (cf. Definition 4.7 on p. 83).

Aleatrati Khosroshahi [AK14, p. 69] further proposes a formal way to determine the complexity of an integration of a single object with the EA repository. However, we claim that with an iterative and incremental approach to Federated EA Model Management, in particular boundary model elements will not be known at an initial stage. Especially, an average number indicating the frequency of boundary model elements will not be available. Measuring complexity upfront to determine and even forecast conflicts that may arise would require considerable efforts providing very little value for EA management. Also, all practitioners confirm that metamodel changes are applied manually and have a considerably impact on the mappings between an EA model and modeling communities, i.e. the mappings might require and adaptation (cf. Section 5.2.3.1).

7.5.5 Model Changes and Conflict Resolution

Aleatrati Khosroshahi [AK14, p. 69–71] elaborates how practitioners describe their processes to adapt metamodels as well as models to changes in their information sources. In particular the import of new information includes an assessment of the quality as well as ‘*overlaps*’ which potentially could be identified with our differencing visualization (cf. Section 5.3.1). To a large extent, any quality assessment—may it be for the metamodel or model—depends on the experience of the enterprise architect or EA repository manager. Another finding of Aleatrati Khosroshahi [AK14, p. 72] confirms our approach to resolve model conflicts. Practitioner state a highly collaborative and lean approach that involves different roles.

Further, a very prominent approach to conflict avoidance for boundary model elements is the introduction of a master-slave relationship between information sources (cf. Definition 5.3).

7.5.6 Additional Factors

In Section 5.2.2.3 we outlined the notion of bidirectional and unidirectional information flow between the federal system and the federated information sources. In our interviews we distinguished between logical and physical flows (cf. human-to-human, human-to-machine, machine-to-human, machine-to-machine in Section 5.2.2.3). Prevalent arguments for unidirectional integration are that the modeling communities want to remain autonomously and bidirectional integration with legacy systems could influence day-to-day business since that are hard to adapt or organizations lack skilled personnel with expertise in the underlying technology [AK14, p. 73]. In contrast, most examples for a bidirectional integration are the propagation of new standards and organization-wide concepts. However, this is a logical bidirectional flow (human-to-human).

Information of a MODELELEMENTS version history is a clear preference of EA experts. Such information serves to analyze the evolution of a MODELELEMENT which sometimes helps to justify decisions and choices made. Besides justification, also tracking of changes for auditing purposes was named as a reason for versioning [AK14, pp. 73–74].

Many EA experts report that EA stakeholders from more business focused modeling communities rarely contribute to EA activities. In contrast Federated EA Model Management requires collaboration among the EA team and all EA stakeholders. Two fundamental approaches are coined by [AK14, pp. 74–75] ‘*social methods*’ and ‘*governance pressure*’; similar concepts can be found in existing literature (cf. Section 3.2.1).

7.5.7 Reflection of the Interview Results

The interview series gave us additional insights in organizations and their processes. Throughout the interviews we gained the impression that most organizations carry out EA model maintenance in an ad hoc manner and individuals rely on (tacit) knowledge to determine the ‘right’ person to resolve conflicts. Further, interviewees had difficulties with our role definitions. These are perceived differently by practitioners—they distinguish by job titles and concrete persons rather than by abstract role definitions. Thus, the proposed roles could not be evaluated. However, the different activities we assigned to the proposed roles (cf. Section 4.1.1 and 5.2), are carried out by an enterprise architect or Domain Expert. Thereby, the latter is an expert of a particular information source.

Identification of an important process step; prior to the interview series this process step has been included in our design rather implicitly. An extensive alignment of terminology is the first step in the integration process (cf. Section 5.2.2.2) and serves as a foundation for the metamodel mapping.

The process designs of Aleatrati Khosroshahi [AK14, p. 65ff] contain several activities that intend to create documentations of models. We intentionally did not include extensive prescriptions for a documentation in Federated EA Model Management since we assume

a modeling system in which changes are tracked by a version history. Since this also holds true for metamodel changes, enterprise architects could document the intent and rationale (cf. also Section 2.1.2) of a change by annotating these changes with a comment. This way, the documentation and actual system are at one place which increases the chance that documentation and model are actually synchronized. In contrast, outdated and possibly extensive documentation in an agile EA management process (cf. Section 2.1.4 and [HRS⁺14]) is counterproductive.

Additional results can be found in a final online survey among 48 EA experts. Aletrati Khosroshahi [AK14, pp. 77–82] could confirm some of our core design decisions, e.g. the involvement of EA stakeholders and other roles in the Federated EA Model Management process, the necessity of a chain of responsibility as well as to follow a lean and pragmatic approach to EA model maintenance.

7.6 Summary

We already drew conclusions of the individual case studies in Section 7.2.4 and in Section 7.3.4 respectively as well as concluding remarks on both case studies (Section 7.4) and the interview series (Section 7.5.7).

Simon [Si96, p. 119] advocates that in actual design situations, it often cannot be proven that solutions are ‘optimal’ and, thus, called ‘best’. However, Simon further introduces his notion of ‘*satisficing*’ solutions as initially introduced in [MS58, p. 140f] and [Si59]. As we currently lack means to find a method to prove an optimal solution, we employ his notion of ‘*satisficing*’ to refer to a good or satisfactory solution.

The evaluation underpins the overall concept of Federated EA Model Management and MODELGLUE seems to provide utility to EA experts. These experts confirmed many aspects of our design. The EA experts not only confirmed the usefulness of our research but also deem our designs as *satisficing* in the sense that it provides utility to them and improves the current situation with respect to their organizational context and personal background. The case studies helped us to reveal concrete design flaws in our prototype. While the first case study had considerable influence on the states of tasks, the second case study had a strong influence on the overall process of Federated EA Model Management. In particular the second case study revealed the importance of the differencing visualization in order to assure model quality prior to initiating the merge process.

In a productive environment, MODELGLUE would have been applied more intensively such that more flaws in our design would have become visible. We foresee that minor adaptations have to be performed on the prototype to reveal the full potential of MODELGLUE to EA stakeholders. However, first adaptations also show that the underlying architecture and implementation is designed in such a way that MODELGLUE can be altered quickly. Many of these proposed and discussed adaptations can be characterized as beyond the purpose of a prototype and lead towards a fully-fledged product. The interview series gave us insights to processes of EA experts, but also pointed out treats to EA management in general and in particular Federated EA Model Management. These social and organizational circumstances have been characterized and also explicated in the form of assumptions for our approach.

In this chapter, we summarize the thesis, reflect on the proposed design based on the research questions raised in Section 1.2, reveal known limitations of our research, and give an overview of further research.

8.1 Summary

The present thesis investigates phenomena that we observed in federated EA model environments. We described typical characteristics of these environments and proposed a solution to address the identified challenges. Throughout the thesis, we detailed a software-supported process that allows to continuously integrate information sources in an EA model. We described a system design that provides means to ensure consistency. Central to our approach is a considerable degree of freedom during model maintenance and collaborative conflict resolution. The former serves to facilitate knowledge management while pursuing the shared goal to restore and maintain CCMC (cf. Definition 5.7 on p. 187). The latter is facilitated by interactive visual support for the communication of model differences as well as model conflicts. Thereby, users can resolve model conflicts interactively. While our previous research addressed a better understanding of the technical and social challenges in a federated EA model environment, the design presented in this thesis is the next step towards a more controlled integration bringing together our findings in a coherent software-supported process. Federated EA Model Management is an iterative and incremental approach to integrate existing information sources with an EA model. We sketched typical process steps, e.g. the selection of relevant information sources and the alignment between different terminologies, and provided a holistic design that we realized as a prototypical software artifact.

Federated EA Model Management responds to demands of practitioners. We found an increasing number of integration efforts in organizations which seek to employ decentralized information sources to maintain a holistic EA model. These attempts were carried out in an ad hoc manner. Neither EA frameworks, literature on EA model maintenance, nor current tool support for EA management provide support for the arising problems and challenges.

Federated EA Model Management is an innovation facilitating the development and maintenance of EA models which make use of existing information within an enterprise. Core contributions of the thesis are (cf. also Section 1.4 and Figure 1.3 on p. 12):

- an extensive description of the **conceptual foundations**, common **use cases**, **requirements**, and **typical characteristics** of a federated EA model environment (A1 – A5),
- an iterative and incremental **process design** and a **system design** (A6) incorporating a **metamodel** (A7) that facilitates its flexible execution,
- formal descriptions of **merge and differencing algorithms** (A8),
- pre-defined and adaptable **conflict resolution strategies** (A9),
- concepts for **interactive visual support** as well as **innovative UI designs** for the collaborative conflict resolution and lessons learned during their implementation (A10 – A13), and
- **feedback from practitioners** who currently pursue Federated EA Model Management (A14 – A16).

In Chapter 1, we described the problem and raised research questions that seek to improve the situation. To build a foundation for the subsequent chapters, we described relevant concepts in EA management, modeling, model merging, and federated database systems in Chapter 2. Thereby, we provided references to chapters and sections in which we applied concepts of other domains that influenced our design. The state-of-the-art was revisited in Chapter 3; with respect to existing literature, we outlined different research gaps identified during our research. Building on the understanding of the state-of-the-art, we described further characteristics of a federated EA model environment in Chapter 4. These characteristics were presented in the context of a deduction of requirements based on the core use cases of Federated EA Model Management. An extensive description of our design was presented in Chapter 5. In Chapter 6, we revealed implementation details. Thereby, we put a particular focus on the challenge of interactive visualizations that were used as a means to support synchronous communication as well as face-to-face support. In the final part of the thesis (Chapter 7), we presented the design and results of a threefold evaluation. We reported on the application of MODELGLUE with industrial data in two case studies whereas an additional interview series among 11 EA experts focused on processes, roles, and other important aspects for an organization pursuing Federated EA Model Management.

After summarizing the core contributions of the present thesis and giving an overview of the individual chapters, we assess our results in the light of the research questions raised in Section 1.2. Thereafter, we provide a mapping of specific sections to the requirements we

derived from the use case analysis and literature in Section 4.3. This mapping reveals to which extent each requirement was addressed.

Research question 1 (RQ1): What are typical characteristics, i.e. core use cases, involved roles, and information sources, of a federated EA model environment?

We diagnosed current problems in Section 1.1 and described the typical situation of a federated EA model environment. Next, we detailed the state-of-the-art in literature in Chapter 3 and provided a topic map that illustrates current research efforts, identified gaps, and challenges in EA model maintenance. Thereby, we sketched technical as well as organizational and social aspects that posed a challenge. Some of these challenges were addressed in the current thesis; however, further research is required that goes beyond efforts underlying a single PhD thesis (cf. Section 8.3). Particulars such as the involved roles and common information sources were extensively described in Section 4.1.1 and 4.1.2. Thereby, we did not only reveal which information sources EA practitioners use but also provided empirical findings on perceived data quality. A formal definition of the relationships between models in a federated EA model environment was illustrated in Section 4.1.3. We described core use cases in Section 4.2 building the foundation for the design of Federated EA Model Management and MODELGLUE.

Research question 2 (RQ2): How can a system design and a process design that continuously integrates models from specialized and autonomous modeling communities with an EA model look like?

In Chapter 5, we presented extensive descriptions of a holistic system design and an iterative process design. Both artifacts seek to evolve an EA model incrementally by integrating new information sources. The process was described in Section 5.2. Central to this process was the collaborative conflict resolution and the continuous adaptation of an organization-specific conflict resolution strategy. As a starting point for organizations, we outlined two pre-defined conflict resolution strategies described in Section 5.2.8. Further, we provided rationales for our design decisions. During the description of the state-of-the-art in EA model maintenance (Chapter 3) and related fields (Chapter 2), we referred to work of others that influenced our design and sketched how we incorporated existing thoughts and principles in a holistic design of Federated EA Model Management. The design of Federated EA Model Management embraces processes and techniques that range from

- an initial selection and integration of information sources into a federation (Section 5.2.2) including the alignment of terminology (Section 5.2.2.2) and the definition of mappings (Section 5.2.2.3 and Section 5.2.2.5), over,
- a triggering-based continuous importing of information to a branch of the respective model in a federal system that served as staging model (Section 5.2.3) as well as
- approaches to model differencing (Section 5.2.4) and merging (Section 5.2.5) as a means for a model synchronization, to

- a description of conflicts between involved models and a process design for the conflict management (Section 5.2.7) as well as adaptable resolution strategies (Section 5.2.8).

Central parts of this software-supported process were facilitated by innovative UI concepts. In Section 5.3 we detailed designs for the visual representation of model differences as well as a conflict management dashboard. The former provided means to ensure model quality by interactively browsing through model and metamodel differences whereas the latter allowed to propagate visual changes to the underlying model and enables collaboration among different parties.

Research question 3 (RQ3): How can a metamodel look like that realizes Federated EA Model Management?

In Section 5.1.2 we introduced a metamodel capable to support Federated EA Model Management. It incorporated a fine-grained access control, responsible roles for model elements, and collaborative state-based tasks which carry conflict information. The role concepts realized a chain of responsibility that allowed to determine the correct addressee of a conflict task. We illustrated essential states of MODELELEMENTS and TASKS and their transitions in Section 5.1.2.5. TASKS build the technical means for enterprise collaboration and facilitate the proposed process (cf. Section 5.2).

Federated EA Model Management can be realized with this metamodel by considering a federation of information sources' models as branches of an EA model (cf. Section 5.2.1). We detailed how to perform differencing (Section 5.2.4) and merging (Section 5.2.5) of instances of this metamodel. Further, we presented concepts to cope with the complexity of an EA model, e.g. a design for an intuitive filter dialog (Section 5.3.1.5) that allowed to define queries for MODELELEMENTS that conform to positive rule sets. In combination with the process, the difference visualization (Section 5.3.1), and the conflict management dashboard (Section 5.3.2), MODELGLUE provided means to monitor data quality within the federation and to restore and maintain CCMC.

Research question 4 (RQ4): What is a suitable integrated UI for collaborative Federated EA Model Management?

In Section 5.3, we described the core concepts of our design for the visual support of MODELGLUE extensively and provided screenshots of our prototypically implemented solution to explain our design. Additional descriptions of our design and screenshots were illustrated in Sections 5.2.6, and 6.2. In Section 5.3.2.5, we described the collaborative aspect of our UI design and its approach to allow to communicate synchronously. We put particular focus on the aspect of real-time collaboration realized through synchronization of visual interactions. Its implementation that was build on the interactive visualization framework (Section 6.3) was sketched in Section 6.4 whereas the respective evaluation results were reported in Section 7.2 and 7.3. Practitioners understood the concepts and especially regarded the navigation through the metamodel of an EA as an intuitive way to browse through an EA model. Compared to expression languages, initial feedback of practitioners confirms that our design for filter dialogs and definitions of rules for the customization of model conflict resolution strategies offers more utility.

Research question 5 (RQ5): What are the software engineering challenges for a system that facilitates Federated EA Model Management?

In Section 4.3 we derived the requirements for MODELGLUE based on the core use cases of Federated EA Model Management. In Chapter 6, we revealed implementation details of MODELGLUE. These included an overview of the component architecture, chosen variability points, screenshots of the prototype, and references to existing literature which describes the implementation of MODELGLUE in more detail. Additionally, we provided insights in the software engineering challenges and their resolutions with respect to interactive visualizations. Therefore, we introduced an extension of a visualization framework that not only allowed to interact with information visually but also enabled the propagation of changes to the underlying model of a visualization. An extensive description of the core concepts and features of the framework can be found in Section 6.3. Further challenges were detailed by the requirements in Section 4.3. These were addressed by the metamodel introduced in Section 5.1.2; e.g. we described the differencing and merging of models formally in Section 5.2.4 and Section 5.2.5 including our notion of equivalence and congruence among different kinds of MODELELEMENTS and techniques for the identification of potentially conflicting changesets. The metamodel built the foundation for our implementation and for the operation-based conflict detection which we explained in Section 5.2.5.6. Further, we described the different kinds of tasks created during conflict detection and merging of models in Section 5.2.5.5.

Research question 6 (RQ6): What is the experience of real stakeholders using this system of systems for EA model maintenance? What are the specificities and further challenges of the Federated EA Model Management process?

In Section 7.2, we presented feedback from EA experts in the insurance industry who agreed to evaluate MODELGLUE with models of their organization. Three different models were combined to a holistic model. In addition, we gave insights in a different case study in Section 7.3 and reported on feedback from a senior enterprise architect who pursued Federated EA Model Management in an organization in the health care domain. In both case studies, the overall process of Federated EA Model Management as well as the software support were evaluated. Thereby, the software support was evaluated with historical live data of the respective organizations. Multiple interviews and workshops served us to find out the specificities of Federated EA Model Management and to iterate our design. While practitioners widely agreed with our major design decisions, their valuable feedback helped refining the design of MODELGLUE. Our design adaptations were sketched in Section 7.2 and Section 7.3.

An additional interview series served to confirm the overall design of Federated EA Model Management. We shed light on the role of a staging model, actual allocation of roles, important process steps of Federated EA Model Management, the notion of approvals by superiors, and other specificities. During the interviews we discussed how to apply our design to identified organization-specific problems. For instance, in Section 7.5, we investigated the role of the master-slave relationship among information sources. In the design of MODELGLUE, we incorporated means to (re-)define an organization-specific

conflict resolution strategy. Due to its adaptable conflict resolution strategy, MODELGLUE can support master-slave strategies between different information sources.

We conclude our summary with a mapping of the developed solution to the requirements derived in Section 4.3. Table 8.1 gives an overview of the requirements, the particular sections in which we describe a solution, and a brief description how we address the requirement indicating to which extent the solution artifact was described in the present thesis.

8.2 Critical Reflection

In Chapter 3, we stated that we do not claim to resolve all issues which were identified and were related to EA model maintenance. In the following, we sketch known limitations of our work; afterwards, we propose directions for further research.

8.2.1 Validity of the Conclusions

Threats to validity of our conclusions and, thus, limitations of our research stem from its evaluation. Although our design includes the concept of OIDs, we had to rely on exact name matching throughout the case study. This could have led to undetected changes and modifications within the information sources.

In our design, we consider all EA stakeholders as target users of MODELGLUE. However, we did not interview any stakeholders outside an EA team. In particular we did not confront business stakeholders with the system.

All results of the case studies must be interpreted with respect to the organizational context and background of the individuals interviewed. The feedback gives insights to opinions of individuals and can be interpreted as confirmation of our design, hints for further improvements, and demands for adaptations when applying MODELGLUE in practice. Given the empirical basis, we cannot generalize our findings at this point. From our perspective, we identified relevant configuration points rather than ultimate changes to our design.

8.2.2 Detection of Changes within Information Sources

Another limitation stems from the absence of active monitors within the information sources. Conceptually, changes in external systems are not monitored and what is actually exchanged between information sources and MODELGLUE strongly depends on the concrete implementation of a physical mapping. This could lead to ‘phantom conflicts tasks’ that are reported within MODELGLUE but are actually already resolved within the information source. This also holds true for deletions within the external system, i.e. the element that is involved in a conflict could cease to exist before the conflict task is opened.

Table 8.1: Mapping of use cases to requirements and respective categories

Req.	Section	Description
PR1	5.1.2, 5.1.2.5	Task support and states
PR2	5.1.2, 5.1.2.4	Determination of responsible roles for MODELELEMENTS
PR3	5.1.2, 5.1.2.2	Responsible roles for all MODELELEMENTS
PR4	5.2	Iterative and incremental process design
Co1	5.2.5, 5.2.5.5, 5.2.6	Conflicts are sent as tasks produced either automatically in the course of a merge or explicitly
Co2	5.1.2	Each task has a designated owner
Co3	5.1.2	Meta-information attached to tasks
Co4	5.1.2, 5.1.2.4	Task forwarding
Co5	5.3.2	Real-time conflict management dashboard
Co6	5.1.2, 6.4	Access control and real time conflict management dashboard
Mo1	5.2.2.5	Ecore-mappings, Excel mappings, and XML-based declarative mappings
Mo2	5.2.3	Import component executes mappings
Mo3	5.1.2	ObjectDefinitions and AttributeDefinition of the metamodel
Mo4	5.2.1, 5.2.3	Model adaptation, model branching, and import component
Mo5	5.1.2	Core feature of the metamodel
Mo6	5.1.2	Constraints are not enforced but violations are captured
Mo7	5.2.4, 5.3.1	Differencing model, algorithm, and visualization
Mo8	5.2.5, 5.2.8	Formal algorithm description
Mo9	5.2.5.6	Formal description of conflict detection
Mo10	5.1.2, 5.1.2.4, 5.2.8, 5.2.6	Conflicts are either detected and assigned by the merge algorithm based on the adaptable conflict resolution strategy or task is created manually.
Mo11	5.1.2	OID concept
Mo12	5.1.2	Constraint violations in branches or EA model
Mo13	5.1.2	Delta-backward changesets
Mo14	5.1.2, 5.1.2.4	Fine-grained access control as well as inheritance of (default) access rights
Us1	5.3, 7.2–7.5	Design, prototype, and feedback from EA experts
Us2	5.3, 7.2–7.5	Design, prototype, and feedback from EA experts
Us3	5.3.1.5	Filter dialog
Us4	5.3.1, 5.3.2	Differencing gives an overview of possible conflicts; interactive conflict management dashboard shows conflicts, conflict tasks can be shown as tabular view
Us5	5.2.8	Adaptable conflict resolution strategy
Us6	5.2.7.1	Learning mechanisms
Us7	5.2.5	Merging does neither affect source nor the target model such that tentative merge results can be reverted
Us8	6.3	Visualizations can be printed on paper with the PNG and PowerPoint renderer
TE1	6.1	The prototype is implemented in an existing web-based platform
TE2	6.1, 5.1.2	Internal data-structure maps to the conceptual metamodel

8.2.3 Modeling Capabilities

MODELGLUE is based on a metamodel which does not include inheritance. Thus, inheritance must be emulated by EA modeling experts. Potentially, the absence of inheritance gives rise to inconsistencies during meta modeling.

Moreover, the metamodel does not feature explicit relationship types. MODELGLUE uses attributes which are interpreted at runtime to describe relationships. This is done to offer a necessary degree of freedom for Federated EA Model Management. However, annotating relationships with additional information is not possible due to this conceptual design.

8.2.4 Implementation Aspects

The prototypical implementation of the collaborative mode within the conflict management dashboard relies on the underlying technology stack; thus, on simultaneous interactions, race-conditions can occur since our design does not embrace locking mechanisms or additional mechanisms that cope with common synchronization challenges.

8.2.5 Non-functional Requirements

Although Kirschner [Ki14] presents initial performance considerations, to a large extent we did not evaluate our prototype concerning non-functional requirements. In particular, performance criteria could become important if the number of instances within an EA model, i.e. OBJECTS and ATTRIBUTES, grows considerably large.

8.3 Further Research

In the final section of this thesis, we discuss further research that could aim to improve the diagnosed situation in EA management. First, we discuss directions directly related to our work. Thereafter, we sketch how the created artifacts could be generalized and applied in a different context.

8.3.1 Additional Case Studies in Real-World Setting

Our evaluation presents initial findings and practitioner feedback that helped us to improve the current prototype. Many enhancements made are subject to be investigated in further research. The participants of the case studies and interview series confirmed the relevance and applicability of the developed artifacts to a large extent but also revealed limitations and further aspects that are relevant for a successful Federated EA Model Management. Suggestions for improvements were incorporated in our design. However, in a series of further case studies, we expect minor changes on the design that may have major impact on its usability. Improvements will not only be limited to the software support of Federated EA Model Management. In particular, the proposed process seeks to describe the core activities. The design of the respective software support is flexible such that other transitions of

activities can be performed by an organization applying our approach. In this vein, the conceptual foundations provided in the present thesis can serve as a frame of reference whereas the software support can be utilized for a continuous monitoring of process variations and behavior during the conflict resolution process. A larger set of case studies could help to refine the process and design of Federated EA Model Management. Especially, long-term observations could reveal patterns for conflict resolution strategies by analyzing common adaptations thereof. These empirical studies could be combined with advanced Artificial Intelligence (AI) techniques to recommend conflict resolution strategies. This could not only improve the learning mechanism proposed in Section 5.2.7.1 but also produce entire conflict resolution strategies based on individual user choices.

8.3.2 Business Cases for Federated EA Model Management

The ROI of EA management is an often discussed topic [Sc05] and, as of today, poses a problem for many EA experts. Although we diagnosed that manual EA model maintenance is an expensive task, we did not provide sufficient evidence for a positive ROI for EA management investments in organizations applying Federated EA Model Management. Empirical data indicates that an automation is desirable, however, there is also qualitative feedback of practitioners that can be interpreted as a counter indicator. Integrating models with a coherent EA model is an expensive task. Against this background, empirical data on ROI considerations would be interesting. Additionally, a broad-scaled research project could seek to identify frequently used information sources. This way, one could develop a catalog of prevalent information sources detailing how these can be combined in a meaningful way to integrate respective information with a core metamodel of an EA. Combined with cost estimations for their integration with an EA model, a cost/benefit analysis could lay grounds for strong EA management business cases.

8.3.3 Standardization of Models and Domain Ontologies for EA Management

Canonical models, standards, and reference architectures for business information systems could help to improve the situation for organizations that seek to maintain their EA model. In particular standards to exchange models and metamodels are of high relevance to exchange not only the model but also the metamodel of an application. Current COTS products implement only rudimentary support for known standards to exchange models as well as metamodels, e.g. EMF. Adhering to one standard would allow to query model and metamodel information via standardized interfaces. In this context, COTS tools could provide utility to create mappings between the different metamodels. One of our assumptions in the present thesis is that a top-down approach which derives domain ontologies from an upper ontology is not suitable for EA management since the information sources already exist and these commonly do not support ontologies. Although ontologies are regarded as too complex by practitioners that are tasked to develop mappings, an interesting research direction is the investigation of the development of new (COTS) IT products that derive their ontologies from an upper ontology. This could further improve the harmonization of models and—ideally—make manual alignments and mappings obsolete.

Further research could investigate the integration of such a canonical development model for business information systems with MODELGLUE. Outcomes could lead towards an upper-ontology for EA management and domain specific ontologies for EA management. An analysis how current APIs look like that satisfy (most) needs of organizations' EA management endeavors could provide a starting point; results could be combined with an investigation on current and future protocols that implement the exchange of models and metamodels with respect to knowledge management.

8.3.4 Usability Experiments

We proposed an innovative design that embraces many aspects and unites elements of

- modeling, e.g. meta modeling and model-to-model transformations,
- software engineering, e.g. architectural design,
- software cartography, e.g. visualizing EA models,
- dashboard design, e.g. layering,
- web-based collaboration, e.g. task management,
- ...

The present thesis provides details of a flexible visualization framework that—to our present understanding—solves most of the software engineering challenges that may arise when adapting the visual concepts to user needs. However, further research could rigorously apply methods common in fields outlined above, e.g. the evaluation methods of the HCI community such as carrying out controlled experiments and observe user behavior to refine the interactive visualizations and the UI design.

8.3.5 Administering Planned States of an EA Model

Focusing on a more general perspective on future activities, further research could apply the outcomes of our research, i.e. the research artifacts, to a different context. For instance, one could apply the solution within the domain of EA management to administer planned states. This could be accomplished by creating planned states as branches of the current state of an EA model. In [Ac13], Achenbach discusses the different relationships of planned states and current states. Some of the planned states (or parts thereof) could be merged into the EA model as soon as transformation projects have been realized. In [RM13], we also investigated how such an evolution could be visualized and sketched arising challenges. Visualizing transformations of the current state with respect to the temporal dimension and merged planned states could be an interesting research topic.

8.3.6 De-contextualizing the Visualizations

The visual concepts presented in this thesis constitute an artifact that could be generalized and applied to other domains. For instance, visual concepts of model differences and conflicts are needed for MDE (cf. Section 2.3.10). Currently, we assume that the investigated models are used for knowledge management only. Further research could bridge the gap between our assumption and the needs of the MDE community. The mapping of EMF to the metamodel of MODELGLUE (cf. Section 5.2.2.5.1) could serve as a starting point. The metamodel then must be extended with additional features of EMF, e.g. methods and inheritance.

8.3.7 De-contextualizing Tasks for Adaptive Case Management

One could de-contextualize the management of task, i.e. applying it to general knowledge management processes. MODELGLUE heavily relies on TASKS and their STATES as well as STATES of MODELELEMENTS. This idea could be applied to general knowledge-intensive processes that are driven by information demands. Literature on adaptive case management [Sw10, SPS11] provides insights to situations during which a facilitation of ad hoc processes is desirable. The presented metamodel could be a starting point for further investigations. One could analyze the demands of other organizational processes and see if the model turns out to be a viable solution for these processes. The metamodel of MODELGLUE could be extended by model events that are triggered if a STATE of a MODELELEMENT or TASK changes.

IMPORT SCHEMA

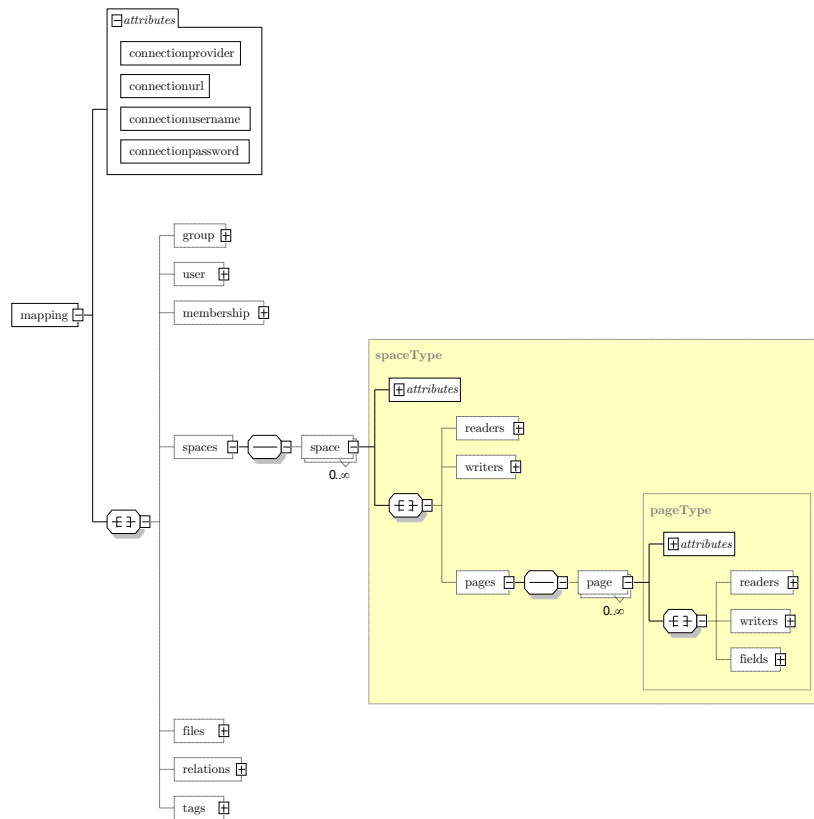


Figure A.1: Core structure of the XML-Schema of a mapping in MODELGLUE

Listing A.1: XML-Schema to specify a mapping for the SQL-based importer

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3   <xs:element name="mapping">
4     <xs:complexType>
5       <xs:all>
6         <xs:element name="group" type="groupType" minOccurs="0"/>
7         <xs:element name="user" type="userType" minOccurs="0"/>
8         <xs:element name="membership" type="membershipType" minOccurs="0"/>
9         <xs:element name="spaces" minOccurs="0">
10          <xs:complexType>
11            <xs:sequence>
12              <xs:element name="space" type="spaceType" minOccurs="0"
13                maxOccurs="unbounded"/>
14            </xs:sequence>
15          </xs:complexType>
16        </xs:element>
17        <xs:element name="files" minOccurs="0">
18          <xs:complexType>
19            <xs:sequence>
20              <xs:element name="file" type="fileType" minOccurs="0"
21                maxOccurs="unbounded"/>
22            </xs:sequence>
23          </xs:complexType>
24        </xs:element>
25        <xs:element name="relations" minOccurs="0">
26          <xs:complexType>
27            <xs:sequence>
28              <xs:element name="relation" type="relationType" minOccurs="0"
29                maxOccurs="unbounded"/>
30            </xs:sequence>
31          </xs:complexType>
32        </xs:element>
33        <xs:element name="tags" minOccurs="0">
34          <xs:complexType>
35            <xs:sequence>
36              <xs:element name="tag" type="tagType" minOccurs="0" maxOccurs="unbounded"/>
37            </xs:sequence>
38          </xs:complexType>
39        </xs:element>
40        <xs:attribute name="connectionprovider" type="xs:string" use="required"/>
41        <xs:attribute name="connectionurl" type="xs:string" use="required"/>
42        <xs:attribute name="connectionusername" type="xs:string" use="required"/>
43        <xs:attribute name="connectionpassword" type="xs:string" use="required"/>
44        <xs:attribute name="documentstoreroor" type="xs:string" use="optional"/>
45      </xs:complexType>
46    </xs:element>
47    <xs:complexType name="spaceType">
48      <xs:all>
49        <xs:element name="readers" minOccurs="0">
50          <xs:complexType>
51            <xs:sequence>
52              <xs:element name="reader" type="principalType" minOccurs="0"
53                maxOccurs="unbounded"/>
54            </xs:sequence>
55          </xs:complexType>
56        </xs:element>
57        <xs:element name="writers" minOccurs="0">
58          <xs:complexType>
59            <xs:sequence>
60              <xs:element name="writer" type="principalType" minOccurs="0"
61                maxOccurs="unbounded"/>
62            </xs:sequence>
63          </xs:complexType>
64        </xs:element>
65        <xs:element name="pages" minOccurs="0">
66          <xs:complexType>
67            <xs:sequence>
68              <xs:element name="page" type="pageType" minOccurs="0" maxOccurs="unbounded"/>
69            </xs:sequence>
70          </xs:complexType>
71        </xs:element>

```



```

68 </xs:all>
69 <xs:attribute name="name" type="xs:string" use="required"/>
70 </xs:complexType>
71 <xs:complexType name="relationType">
72 <xs:attribute name="tablename" type="xs:string" use="required"/>
73 <xs:attribute name="tricianame" type="xs:string" use="required"/>
74 <xs:attribute name="fromkeycolumnname" type="xs:string" use="required"/>
75 <xs:attribute name="fromparent" type="xs:string" use="required"/>
76 <xs:attribute name="tokeycolumnname" type="xs:string" use="required"/>
77 <xs:attribute name="toparent" type="xs:string" use="required"/>
78 <xs:attribute name="filter" type="xs:string" use="optional"/>
79 </xs:complexType>
80 <xs:complexType name="tagType">
81 <xs:attribute name="tablename" type="xs:string" use="required"/>
82 <xs:attribute name="tricianame" type="xs:string" use="required"/>
83 <xs:attribute name="idfield" type="xs:string" use="required"/>
84 <xs:attribute name="tagfield" type="xs:string" use="required"/>
85 <xs:attribute name="parent" type="xs:string" use="required"/>
86 <xs:attribute name="filter" type="xs:string" use="optional"/>
87 </xs:complexType>
88 <xs:complexType name="pageType">
89 <xs:all>
90 <xs:element name="readers" minOccurs="0">
91 <xs:complexType>
92 <xs:sequence>
93 <xs:element name="reader" type="principalType" minOccurs="0"
94 </xs:sequence>
95 </xs:complexType>
96 </xs:element>
97 <xs:element name="writers" minOccurs="0">
98 <xs:complexType>
99 <xs:sequence>
100 <xs:element name="writer" type="principalType" minOccurs="0"
101 </xs:sequence>
102 </xs:complexType>
103 </xs:element>
104 <xs:element name="fields" minOccurs="0">
105 <xs:complexType>
106 <xs:sequence>
107 <xs:element name="field" type="fieldType" minOccurs="0"
108 </xs:sequence>
109 </xs:complexType>
110 </xs:element>
111 </xs:all>
112 <xs:attribute name="tablename" type="xs:string" use="required"/>
113 <xs:attribute name="tricianame" type="xs:string" use="required"/>
114 <xs:attribute name="namefield" type="xs:string" use="required"/>
115 <xs:attribute name="idfield" type="xs:string" use="required"/>
116 <xs:attribute name="filter" type="xs:string" use="optional"/>
117 <xs:attribute name="contentfield" type="xs:string" use="optional"/>
118 <xs:attribute name="lasteditorfield" type="xs:string" use="optional"/>
119 <xs:attribute name="lastmodificationtimestampfield" type="xs:string"
120 <xs:attribute name="parentpagefield" type="xs:string" use="optional"/>
121 <xs:attribute name="parentpageentitytype" type="xs:string" use="optional"/>
122 <xs:attribute name="order" type="xs:string" use="optional"/>
123 <xs:attribute name="idprefix" type="xs:string" use="optional"/>
124 <xs:attribute name="usedefaultparentpage" type="xs:boolean" use="optional"
125 </xs:complexType>
126 <xs:complexType name="fileType">
127 <xs:all>
128 <xs:element name="readers" minOccurs="0">
129 <xs:complexType>
130 <xs:sequence>
131 <xs:element name="reader" type="principalType" minOccurs="0"
132 </xs:sequence>
133 </xs:complexType>
134 </xs:element>
135 <xs:element name="writers" minOccurs="0">

```

A. Import Schema

```
136 <xs:complexType>
137 <xs:sequence>
138 <xs:element name="writer" type="principalType" minOccurs="0"
    maxOccurs="unbounded"/>
139 </xs:sequence>
140 </xs:complexType>
141 </xs:element>
142 <xs:element name="fields" minOccurs="0">
143 <xs:complexType>
144 <xs:sequence>
145 <xs:element name="field" type="fieldType" minOccurs="0"
    maxOccurs="unbounded"/>
146 </xs:sequence>
147 </xs:complexType>
148 </xs:element>
149 </xs:all>
150 <xs:attribute name="tablename" type="xs:string" use="required"/>
151 <xs:attribute name="tricianame" type="xs:string" use="required"/>
152 <xs:attribute name="namefield" type="xs:string" use="required"/>
153 <xs:attribute name="idfield" type="xs:string" use="required"/>
154 <xs:attribute name="filepathfield" type="xs:string" use="required"/>
155 <xs:attribute name="directorylookupquery" type="xs:string" use="optional"/>
156 <xs:attribute name="filter" type="xs:string" use="optional"/>
157 <xs:attribute name="contentfield" type="xs:string" use="optional"/>
158 <xs:attribute name="lasteditorfield" type="xs:string" use="optional"/>
159 <xs:attribute name="lastmodificationtimestampfield" type="xs:string"
    use="optional"/>
160 <xs:attribute name="parentpagefield" type="xs:string" use="optional"/>
161 <xs:attribute name="parentpageentitytype" type="xs:string" use="optional"/>
162 <xs:attribute name="order" type="xs:string" use="optional"/>
163 <xs:attribute name="idprefix" type="xs:string" use="optional"/>
164 </xs:complexType>
165 <xs:complexType name="principalType">
166 <xs:attribute name="type" use="required">
167 <xs:simpleType>
168 <xs:restriction base="xs:string">
169 <xs:enumeration value="groupname"/>
170 <xs:enumeration value="username"/>
171 <xs:enumeration value="groupid"/>
172 <xs:enumeration value="userid"/>
173 <xs:enumeration value="system"/>
174 </xs:restriction>
175 </xs:simpleType>
176 </xs:attribute>
177 <xs:attribute name="isderived" type="xs:boolean" use="optional" default="false"/>
178 <xs:attribute name="value" type="xs:string" use="required"/>
179 </xs:complexType>
180 <xs:complexType name="fieldType">
181 <xs:attribute name="columnname" type="xs:string" use="required"/>
182 <xs:attribute name="tricianame" type="xs:string" use="required"/>
183 <xs:attribute name="delimiter" type="xs:string" use="optional" default=";"/>
184 <xs:attribute name="triciatype" use="optional" default="Text">
185 <xs:simpleType>
186 <xs:restriction base="xs:string">
187 <xs:enumeration value="Text"/>
188 <xs:enumeration value="Link"/>
189 <xs:enumeration value="MultiText"/>
190 <xs:enumeration value="Number"/>
191 <xs:enumeration value="Date"/>
192 </xs:restriction>
193 </xs:simpleType>
194 </xs:attribute>
195 <xs:attribute name="parent" type="xs:string" use="optional"/>
196 <xs:attribute name="filter" type="xs:string" use="optional"/>
197 <xs:attribute name="importifnullorempty" type="xs:boolean" use="optional"
    default="false"/>
198 </xs:complexType>
199 <xs:complexType name="groupType">
200 <xs:attribute name="tablename" type="xs:string" use="required"/>
201 <xs:attribute name="namefield" type="xs:string" use="required"/>
202 <xs:attribute name="idfield" type="xs:string" use="required"/>
203 <xs:attribute name="contentfield" type="xs:string" use="optional"/>
204 <xs:attribute name="filter" type="xs:string" use="optional"/>
205 <xs:attribute name="order" type="xs:string" use="optional"/>
```

```

206 <xs:attribute name="idprefix" type="xs:string" use="optional"/>
207 </xs:complexType>
208 <xs:complexType name="userType">
209 <xs:all>
210 <xs:element name="fields" minOccurs="0">
211 <xs:complexType>
212 <xs:sequence>
213 <xs:element name="field" type="fieldType" minOccurs="0"
    maxOccurs="unbounded"/>
214 </xs:sequence>
215 </xs:complexType>
216 </xs:element>
217 </xs:all>
218 <xs:attribute name="tablename" type="xs:string" use="required"/>
219 <xs:attribute name="namefield" type="xs:string" use="required"/>
220 <xs:attribute name="idfield" type="xs:string" use="required"/>
221 <xs:attribute name="loginfield" type="xs:string" use="required"/>
222 <xs:attribute name="passwordfield" type="xs:string" use="required"/>
223 <xs:attribute name="contentfield" type="xs:string" use="optional"/>
224 <xs:attribute name="filter" type="xs:string" use="optional"/>
225 <xs:attribute name="order" type="xs:string" use="optional"/>
226 <xs:attribute name="idprefix" type="xs:string" use="optional"/>
227 </xs:complexType>
228 <xs:complexType name="membershipType">
229 <xs:attribute name="tablename" type="xs:string" use="required"/>
230 <xs:attribute name="groupfield" type="xs:string" use="required"/>
231 <xs:attribute name="userfield" type="xs:string" use="required"/>
232 <xs:attribute name="groupfieldtype" use="optional" default="id">
233 <xs:simpleType>
234 <xs:restriction base="xs:string">
235 <xs:enumeration value="name"/>
236 <xs:enumeration value="id"/>
237 </xs:restriction>
238 </xs:simpleType>
239 </xs:attribute>
240 <xs:attribute name="userfieldtype" use="optional" default="id">
241 <xs:simpleType>
242 <xs:restriction base="xs:string">
243 <xs:enumeration value="name"/>
244 <xs:enumeration value="id"/>
245 </xs:restriction>
246 </xs:simpleType>
247 </xs:attribute>
248 <xs:attribute name="filter" type="xs:string" use="optional"/>
249 </xs:complexType>
250 </xs:schema>

```

APPENDIX B

MAPPING OF THE CONCEPTUAL MODEL TO THE
IMPLEMENTATION

B.1 ModelElements and Roles

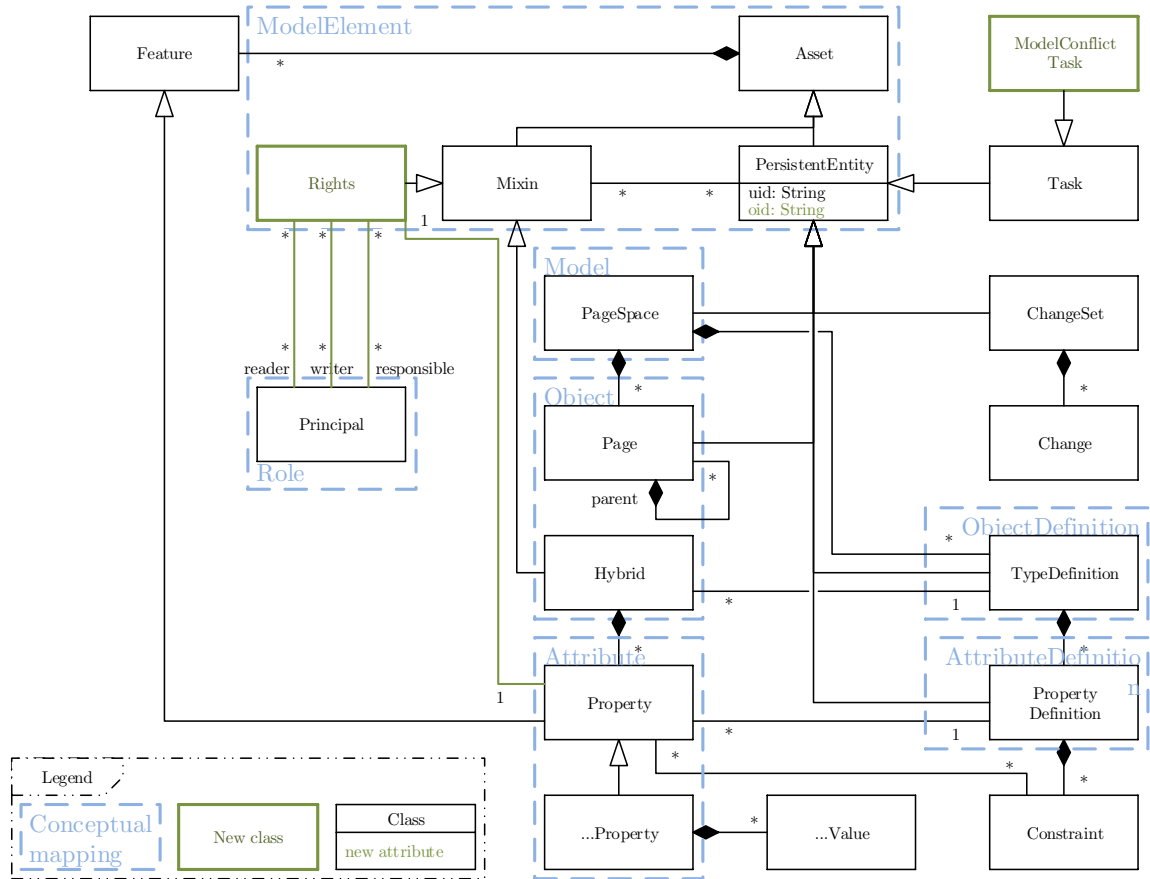


Figure B.1: Mapping of MODEL ELEMENTS and ROLES to the prototypical implementation [Ki14, p. 21]

B.2 Changesets and Operations

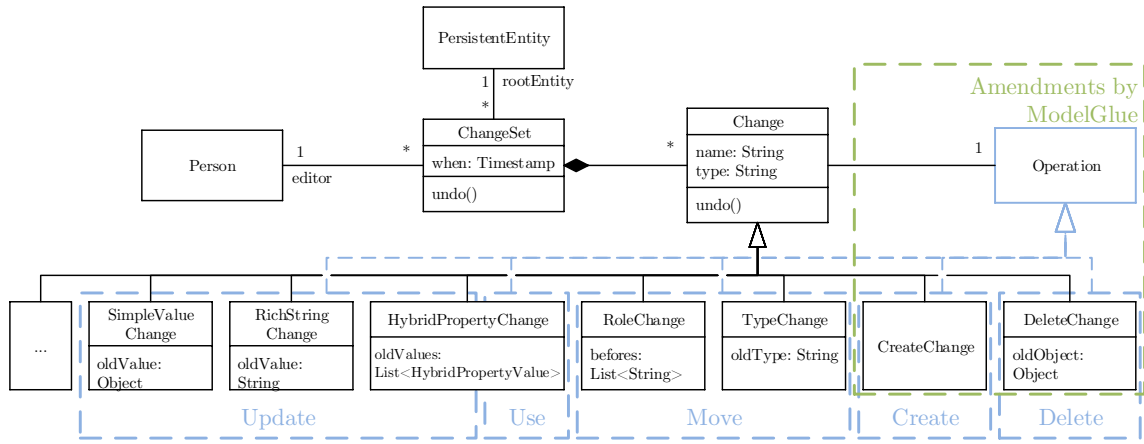


Figure B.2: Mapping of OPERATIONS to the prototypical implementation [Ki14, p. 34]

Bibliography

- [ABB⁺07] Arbab, F.; Boer, F. S. d.; Bonsangue, M. M.; Lankhorst, M. M.; Proper, E. H.; Torre, L. W. N. v. d.: *Integrating Architectural Models - Symbolic, Semantic and Subjective Models in Enterprise Architecture*. *Enterprise Modelling and Information Systems Architectures*. 2(1):40–57. 2007.
- [ABP06] Adams, R.; Bessant, J.; Phelps, R.: *Innovation management measurement: A review*. *International Journal of Management Reviews*. 8(1):21–47. 2006.
- [Ac13] Achenbach, P.: *Framework for the strategic planning of enterprise architectures*. Master's thesis. Technical University Munich. 2013.
- [AH98] Arkin, E. M.; Hassin, R.: *On local search for weighted k-set packing*. *Mathematics of Operations Research*. 23(3):640–648. 1998.
- [AHR14] Aleatrati, P.; Hauder, M.; Roth, S.: *Impact of Solvency II on the Enterprise Architecture of Insurances: A Qualitative Study in Germany*. In *Multikonferenz Wirtschaftsinformatik (MKWI)*. Paderborn, Germany. 2014. best student paper award.
- [Ai13] Aier, S.: *Understanding the Role of Organizational Culture for Design and Success of Enterprise Architecture Management*. In *11th International Conference on Wirtschaftsinformatik (WI), Leipzig, Germany*. 2013.
- [AK01a] Armour, F. J.; Kaisler, S. H.: *Enterprise architecture: Agile transition and implementation*. *IT professional*. 3(6):30–37. 2001.
- [AK01b] Atkinson, C.; Kühne, T.: *The essence of multilevel metamodeling*. In *«UML» 2001—The Unified Modeling Language. Modeling Languages, Concepts, and Tools*. pages 19–33. Springer. 2001.
- [AK03] Atkinson, C.; Kühne, T.: *Model-driven development: a metamodeling foundation*. *Software, IEEE*. 20(5):36–41. 2003.

- [AK14] Aletrati Khosroshahi, P.: *Evaluation of Governance- and Process Structures of the Federated Enterprise Architecture Model Management*. Master's thesis. Technical University Munich. 2014.
- [AKL99a] Armour, F. J.; Kaisler, S. H.; Liu, S. Y.: *A big-picture look at Enterprise Architectures*. *IT professional*. 1(1):35–42. 1999.
- [AKL99b] Armour, F. J.; Kaisler, S. H.; Liu, S. Y.: *Building an enterprise architecture step by step*. *IT professional*. 1(4):31–39. 1999.
- [AKS⁺08] Aier, S.; Kurpjuweit, S.; Schmitz, O.; Schulz, J.; Thomas, A.; Winter, R.: *An Engineering Approach to Enterprise Architecture Design and its Application at a Financial Service Provider*. In *Modellierung betrieblicher Informationssysteme (MobIS 2008) – Modellierung zwischen SOA und Compliance Management 27.-28. November 2008 Saarbrücken*. pages 115–130. 2008.
- [Al14] Altova: *MapForce — Graphical Data Mapping, Conversion, and Integration Tool*. 2014. . Online <http://www.altova.com/mapforce.html> (last accessed: Saturday 1st August, 2015).
- [ANO09] Austin, R. D.; Nolan, R. L.; O'Donnell, S.: *The adventures of an IT leader*. Harvard Business Press. 2009.
- [ANV05] Ambler, S.; Nalbone, J.; Vizdos, M.: *Enterprise unified process, the: extending the rational unified process*. Prentice Hall Press. 2005.
- [AP03] Alanen, M.; Porres, I.: *Difference and Union of Models*. In (Stevens, P.; Whittle, J.; Booch, G., Ed.): *«UML» 2003 - The Unified Modeling Language. Modeling Languages and Applications*. volume 2863 of *Lecture Notes in Computer Science*. pages 2–17. Springer Berlin Heidelberg. 2003.
- [Ap14] Apache Foundation: *Subversion Release Notes*. 2014. . Online <http://subversion.apache.org/docs/release-notes/> (last accessed: Saturday 1st August, 2015).
- [As94] Asklund, U.: *Identifying Conflicts During Structural Merge*. In *IN MAGNUS-SON ET AL. MHM94*. pages 1–3. 1994.
- [ASM⁺12] Ahlemann, F.; Stettiner, E.; Messerschmidt, M.; Legner, C.: *Strategic enterprise architecture management: challenges, best practices, and future developments*. Springer. 2012.
- [ASW09] Altmanninger, K.; Seidl, M.; Wimmer, M.: *A survey on model versioning approaches*. *International Journal of Web Information Systems*. 5(3):271–304. 2009.
- [AV10] Alegria, A.; Vasconcelos, A.: *IT Architecture automatic verification: A network evidence-based approach*. In *Research Challenges in Information Science (RCIS), 2010 Fourth International Conference on*. pages 1–12. May 2010.
- [Ba68] Bagley, P. R.: *Extension of Programming Language Concepts*. National Bureau of Standards, Institute for Applied Technology. 1968.

- [Ba96] Basili, V. R.: *The role of experimentation in software engineering: past, current, and future*. In *Proceedings of the 18th international conference on software engineering*. pages 442–449. IEEE Computer Society. 1996.
- [Ba08] Bartelt, C.: *Consistence preserving model merge in collaborative development processes*. In *Proceedings of the 2008 international workshop on Comparison and versioning of software models*. pages 13–18. ACM. 2008.
- [Ba13] Baranovskiy, D.: *Raphaël—JavaScript Library*. 2013. . Online <http://raphaeljs.com> (last accessed: Saturday 1st August, 2015).
- [BBK⁺13] Binz, T.; Breitenbucher, U.; Kopp, O.; Leymann, F.: *Automated Discovery and Maintenance of Enterprise Topology Graphs*. In *Service-Oriented Computing and Applications (SOCA), 2013 IEEE 6th International Conference on*. pages 126–134. Dec 2013.
- [BBL12] Bente, S.; Bombosch, U.; Langade, S.: *Collaborative Enterprise Architecture: Enriching EA with lean, agile, and enterprise 2.0 practices*. Newnes. 2012.
- [BCE⁺06] Brunet, G.; Chechik, M.; Easterbrook, S.; Nejati, S.; Niu, N.; Sabetzadeh, M.: *A manifesto for model merging*. In *Proceedings of the 2006 international workshop on Global integrated model management*. pages 5–12. ACM. 2006.
- [BCF⁺09] Batini, C.; Cappiello, C.; Francalanci, C.; Maurino, A.: *Methodologies for data quality assessment and improvement*. *ACM Computing Surveys (CSUR)*. 41(3):16. 2009.
- [BCM10] Bughin, J.; Chui, M.; Manyika, J.: *Clouds, big data, and smart assets: Ten tech-enabled business trends to watch*. *McKinsey Quarterly*. 56(1):75–86. 2010.
- [BCV99] Bergamaschi, S.; Castano, S.; Vincini, M.: *Semantic integration of semistructured and structured data sources*. *ACM Sigmod Record*. 28(1):54–59. 1999.
- [BD99] Bruegge, B.; Dutoit, A. A.: *Object-Oriented Software Engineering; Conquering Complex and Changing Systems*. Prentice Hall PTR. Upper Saddle River, NJ, USA. 1999.
- [BDM⁺10] Buckl, S.; Dierl, T.; Matthes, F.; Schweda, C. M.: *Building Blocks for Enterprise Architecture Management Solutions*. In *2nd Practice-driven Research on Enterprise Transformation (PRET) working conference*. Delft, NL. 2010.
- [Be68] von Bertalanffy, L.: *General system theory: foundations, development, applications*. International library of systems theory and philosophy. G. Braziller. 1968.
- [Be04] Bergsten, H.: *JavaServer faces*. O’Reilly Media, Inc. 2004.
- [Be12] Belbin, R. M.: *Team Roles at Work*. Taylor & Francis. 2nd edition. 2012.
- [BEG⁺12] Buschle, M.; Ekstedt, M.; Grunow, S.; Hauder, M.; Matthes, F.; Roth, S.: *Automated Enterprise Architecture Documentation using an Enterprise Service Bus*. In *Proceedings of in Americas conference on Information Systems (AMCIS)*. 2012.

- [BEL⁺08] Buckl, S.; Ernst, A. M.; Lankes, J.; Matthes, F.: *Enterprise Architecture Management Pattern Catalog (Version 1.0, February 2008)*. Technical report. Technical University Munich. Munich, Germany. 2008.
- [BG13] Bauer, A.; Günzel, H.: *Data-Warehouse-Systeme*. dpunkt.verlag. 4th edition. 2013.
- [BGP11] Brückmann, T.; Gruhn, V.; Pfeiffer, M.: *Towards real-time monitoring and controlling of enterprise architectures using business software control centers*. In *Software Architecture*. pages 287–294. Springer. 2011.
- [BH09] Bostock, M.; Heer, J.: *Protovis: a graphical toolkit for visualization*. *IEEE Transactions on Visualization and Computer Graphics*. 15(6):1121–1128. 2009.
- [BHG87] Bernstein, P. A.; Hadzilacos, V.; Goodman, N.: *Concurrency Control and Recovery in Database Systems*. Addison-Wesley. 1987.
- [BHS⁺12] Buschle, M.; Holm, H.; Sommestad, T.; Ekstedt, M.; Shahzad, K.: *A Tool for automatic Enterprise Architecture modeling*. In *IS Olympics: Information Systems in a Diverse World*. pages 1–15. Springer. 2012.
- [BKL00] Busse, S.; Kutsche, R.-D.; Leser, U.: *Strategies for the Conceptual Design of Federated Information Systems*. In *EFIS*. pages 23–32. 2000.
- [BKL⁺99] Busse, S.; Kutsche, R.-D.; Leser, U.; Weber, H.: *Federated Information Systems: Concepts, Terminology and Architectures*. Technical report. TU Berlin. 1999. TB Nr. 99-9.
- [BLFM05] Berners-Lee, T.; Fielding, R.; Masinter, L.: *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986 (INTERNET STANDARD). January 2005.
- [BLHL01] Berners-Lee, T.; Hendler, J.; Lassila, O.: *The semantic web*. *Scientific American*. 284(5):28–37. 2001.
- [BLS⁺10] Brosch, P.; Langer, P.; Seidl, M.; Wieland, K.; Wimmer, M.; Kappel, G.: *Concurrent modeling in early phases of the software development life cycle*. In *Proceedings of the 16th international conference on Collaboration and technology*. CRIWG'10. pages 129–144. Berlin, Heidelberg. 2010. Springer-Verlag.
- [BM07] Bernstein, P. A.; Melnik, S.: *Model Management 2.0: Manipulating Richer Mappings*. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. SIGMOD '07. pages 1–12. New York, NY, USA. 2007. ACM.
- [BMM⁺08] Blanc, X.; Mounier, I.; Mougnot, A.; Mens, T.: *Detecting model inconsistency through operation-based model construction*. In *ACM/IEEE 30th International Conference on Software Engineering (ICSE)*. pages 511–520. IEEE. 2008.
- [BMM⁺11a] Buckl, S.; Matthes, F.; Monahov, I.; Roth, S.; Schulz, C.; M., S. C.: *Towards an Agile Design of the Enterprise Architecture Management Function*. In *Trends in Enterprise Architecture Research (TEAR)*. 2011.

-
- [BMM⁺11b] Buckl, S.; Matthes, F.; Monahov, I.; Roth, S.; Schulz, C.; Schweda, C. M.: *Enterprise Architecture Management Patterns for Enterprise-wide Access Views on Business Objects*. In *European Conference on Pattern Languages of Programs (EuroPLOP) 2011*. Irsee Monastery, Bavaria, Germany. 2011.
- [BMN⁺11] Buckl, S.; Matthes, F.; Neubert, C.; Schweda, C. M.: *A lightweight approach to enterprise architecture modeling and documentation*. In *Information Systems Evolution*. pages 136–149. Springer. 2011.
- [BMR⁺10a] Buckl, S.; Matthes, F.; Roth, S.; Schulz, C.; Schweda, C. M.: *A Conceptual Framework for Enterprise Architecture Design*. In *Workshop on Trends in Enterprise Architecture Research (TEAR)*. Delft, The Netherlands. 2010.
- [BMR⁺10b] Buckl, S.; Matthes, F.; Roth, S.; Schulz, C.; Schweda, C. M.: *A method for constructing enterprise-wide access views on business objects*. In *Informatik 2010: IT-Governance in verteilten Systemen (GVS 2010)*. Leipzig, Germany. 2010.
- [Bo53] Boole, G.: *An Investigation of the Laws of Thought: On which are Founded the Mathematical Theories of Logic and Probabilities*. Walton and Maberly. 1853.
- [Bo97] Bordia, P.: *Face-to-face versus computer-mediated communication: A synthesis of the experimental literature*. *Journal of Business Communication*. 34(1):99–118. 1997.
- [BOH11] Bostock, M.; Ogievetsky, V.; Heer, J.: *D³: Data-Driven Documents*. *IEEE Transactions on Visualization and Computer Graphics*. 17(12):2301–2309. December 2011.
- [BP08] Brun, C.; Pierantonio, A.: *Model differences in the eclipse modeling framework*. *UPGRADE, The European Journal for the Informatics Professional*. 9(2):29–34. 2008.
- [Br96] Brinkkemper, S.: *Method engineering: engineering of information systems development methods and tools*. *Information and software technology*. 38(4):275–280. 1996.
- [Br97] Bradner, S.: *Key words for use in RFCs to Indicate Requirement Levels*. RFC 2119 (Best Current Practice). March 1997.
- [Br12] Brosch, P.: *Conflict Resolution in Model Versioning*. PhD thesis. Vienna University of Technology. 2012.
- [Br13] Bruegge, B.: *Creativity vs Rigor: Informal Modeling is OK*. 2013. Keynote presentation at MODELS.
- [BS81] Bancilhon, F.; Spyratos, N.: *Update semantics of relational views*. *ACM Transactions on Database Systems (TODS)*. 6(4):557–575. 1981.

- [BSH86] Basili, V. R.; Selby, R. W.; Hutchens, D. H.: *Experimentation in Software Engineering*. *IEEE Transactions on Software Engineering*. 12(7):733–743. 1986.
- [BSH98] Brinkkemper, S.; Saeki, M.; Harmsen, F.: *Assembly techniques for method engineering*. In *Advanced Information Systems Engineering*. pages 381–400. Springer. 1998.
- [BSW⁺09] Brosch, P.; Seidl, M.; Wieland, K.; Wimmer, M.; Langer, P.: *We can work it out: Collaborative Conflict Resolution in Model Versioning*. In (Wagner, I.; Tellioglu, H.; Balka, E.; Simone, C.; Ciolfi, L., Ed.): *ECSCW 2009*. pages 207–214. Springer London. 2009.
- [Bu95] Buffenbarger, J.: *Syntactic software merging*. In (Estublier, J., Ed.): *Software Configuration Management*. volume 1005 of *Lecture Notes in Computer Science*. pages 153–172. Springer Berlin Heidelberg. 1995.
- [Bü07] Büchner, T.: *Introspektive modellgetriebene Softwareentwicklung*. PhD thesis. Fakultät für Informatik, Technische Universität München. 2007.
- [Bu11] Buckl, S.: *Developing organization-specific enterprise architecture management functions using a method base*. PhD thesis. Technical University Munich. 2011.
- [Ca11a] Cabinet Office: *ITIL Continual Service Improvement*. The Stationery Office. Norwich, UK. 2011.
- [Ca11b] Cabinet Office: *ITIL Lifecycle Suite 2011 Edition*. The Stationery Office. Norwich, UK. 2011.
- [Ca11c] Cabinet Office: *ITIL Service Design*. The Stationery Office. Norwich, UK. 2011.
- [Ca11d] Cabinet Office: *ITIL Service Operation*. The Stationery Office. Norwich, UK. 2011.
- [Ca11e] Cabinet Office: *ITIL Service Strategy*. The Stationery Office. Norwich, UK. 2011.
- [Ca11f] Cabinet Office: *ITIL Service Transition*. The Stationery Office. Norwich, UK. 2011.
- [CDRP08] Cicchetti, A.; Di Ruscio, D.; Pierantonio, A.: *Managing model conflicts in distributed development*. In *Model Driven Engineering Languages and Systems*. pages 311–325. Springer. 2008.
- [CG02] Cecconi, A.; Galanda, M.: *Adaptive Zooming in Web Cartography*. *Computer Graphics Forum*. 21(4):787–799. 2002.
- [CGT75] Chamberlin, D. D.; Gray, J.; Traiger, I. L.: *Views, authorization, and locking in a relational data base system*. In *Proceedings of the national computer conference and exposition*. pages 425–430. ACM. 1975.

-
- [Ch76] Chen, P. P.-S.: *The Entity-Relationship Model – Toward a Unified View of Data*. *ACM Transactions on Database Systems (TODS)*. 1(1):9–36. 1976.
- [CH03] Czarnecki, K.; Helsen, S.: *Classification of model transformation approaches*. In *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*. volume 45. pages 1–17. 2003.
- [Ch08] Chacon, S.: *Git Internals*. Peepcode. 2008.
- [Ch09] Chacon, S.: *Pro git*. Apress. 2009.
- [CHL⁺13] Chen, W.; Hess, C.; Langermeier, M.; Stuelpnagel, J.; Diefenthaler, P.: *Semantic Enterprise Architecture Management*. In *Proceedings of the 15th International Conference on Enterprise Information Systems (ICEIS)*. 2013.
- [Ci14] Cisco: *Cisco WebEx Web Conferencing, Online Meetings, Desktop Sharing, Video Conferencing*. 2014. . Online <http://www.webex.com> (last accessed: Saturday 1st August, 2015).
- [CJB99] Chandrasekaran, B.; Josephson, J. R.; Benjamins, V. R.: *What are ontologies, and why do we need them? Intelligent Systems and Their Applications, IEEE*. 14(1):20–26. 1999.
- [CMS99] Card, S. K.; Mackinlay, J. D.; Shneiderman, B.: *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers. 1999.
- [Co97] Conrad, S.: *Föderierte Datenbanksysteme: Konzepte der Datenintegration*. Springer. 1997.
- [Co01] Cockburn, A.: *Writing effective use cases*. Addison-Wesley. 2001.
- [CSH06] Choi, N.; Song, I.-Y.; Han, H.: *A survey on ontology mapping*. *ACM Sigmod Record*. 35(3):34–41. 2006.
- [CW98] Conradi, R.; Westfechtel, B.: *Version models for software configuration management*. *ACM Computing Surveys (CSUR)*. 30(2):232–282. June 1998.
- [Da04] Date, C. J.: *An Introduction To Database Systems*. Pearson Education India. 8th edition. 2004.
- [Da05] Davenport, T. H.: *Thinking for a living: how to get better performances and results from knowledge workers*. Harvard Business Press. 2005.
- [DB78] Dayal, U.; Bernstein, P. A.: *On the Updatability of Relational Views*. In *VLDB*. volume 78. pages 368–377. 1978.
- [De82] Deming, E. W.: *Out of the Crisis*. Massachusetts Institute of Technology. Cambridge, MA, USA. 1982.
- [De02] Devedzić, V.: *Understanding ontological engineering*. *Communications of the ACM*. 45(4):136–144. 2002.

- [De09] Department of Defense (DoD) USA: *DoD Architecture Framework Version 2.0: Volume 1: Introduction, Overview, and Concepts – Manager’s Guide*. 2009.
- [DGS⁺09] Doucet, G.; Götzt, J.; Saha, P.; Bernard, S.: *Coherency Management: Architecting the Enterprise for Alignment, Agility and Assurance*. AuthorHouse. Blommington, IN, USA. 2009.
- [DH84] Dayal, U.; Hwang, H.-Y.: *View definition and generalization for database integration in a multidatabase system*. *Software Engineering, IEEE Transactions on*. (6):628–645. 1984.
- [DHM10] Davenport, T. H.; Harris, J. G.; Morison, R.: *Analytics at work: Smarter decisions, better results*. Harvard Business Press. 2010.
- [DL04] ter Doest, H.; Lankhorst, M.: *Tool Support for Enterprise Architecture-A Vision*. Technical report. Telematica Instituut, Enschede. 2004.
- [DMJ⁺07] Dig, D.; Manzoor, K.; Johnson, R.; Nguyen, T. N.: *Refactoring-aware configuration management for object-oriented programs*. In *29th International Conference on Software Engineering (ICSE)*. pages 427–436. IEEE. 2007.
- [DS90] Davenport, T. H.; Short, J. E.: *The new industrial engineering: information technology and business process redesign*. *Sloan management review*. 31(4). 1990.
- [DS05] Duerst, M.; Suignard, M.: *Internationalized Resource Identifiers (IRIs)*. RFC 3987 (Proposed Standard). January 2005.
- [EAB10] Espinosa, J. A.; Armour, F.; Boh, W. F.: *Coordination in enterprise architecting: an interview study*. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*. pages 1–10. IEEE. 2010.
- [Ed97] Edwards, W. K.: *Flexible conflict detection and management in collaborative applications*. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*. pages 139–148. ACM. 1997.
- [EHH⁺08] Engels, G.; Hess, A.; Humm, B.; Juwig, O.; Lohmann, M.; Richter, J.-P.: *Quasar Enterprise – Anwendungslandschaften serviceorientiert gestalten*. dpunkt.verlag. Heidelberg, Germany. 2008.
- [Ev04] Evans, E.: *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional. 2004.
- [FA13] Fahland, D.; van der Aalst, W. M.: *Model repair—aligning process models to reality*. *Information Systems*. 2013.
- [FAB⁺11a] Farwick, M.; Agreiter, B.; Breu, R.; Ryll, S.; Voges, K.; Hanschke, I.: *Automation processes for enterprise architecture management*. In *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2011 15th IEEE International*. pages 340–349. IEEE. 2011.

- [FAB⁺11b] Farwick, M.; Agreiter, B.; Breu, R.; Ryll, S.; Voges, K.; Hanschke, I.: *Requirements for Automated Enterprise Architecture Model Maintenance - A Requirements Analysis based on a Literature Review and an Exploratory Survey*. In *Proceedings of the 13th International Conference on Enterprise Information Systems (ICEIS)*. pages 325–337. SciTePress. 2011.
- [FAW07] Fischer, R.; Aier, S.; Winter, R.: *A Federated Approach to Enterprise Architecture Model Maintenance*. In *Enterprise Modelling and Information Systems Architectures – Concepts and Applications, Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2007), St. Goar, Germany, October 8-9, 2007*. pages 9–22. St. Goar, Germany. 2007.
- [FBH⁺13] Farwick, M.; Breu, R.; Hauder, M.; Roth, S.; Matthes, F.: *Enterprise Architecture Documentation: Empirical Analysis of Information Sources for Automation*. In *46th Hawaii International Conference on System Sciences (HICSS), Maui, Hawaii*. 2013.
- [Fe89] Feather, M. S.: *Detecting interference when merging specification evolutions*. In *ACM SIGSOFT Software Engineering Notes*. volume 14. pages 169–176. ACM. 1989.
- [Fe06] Few, S.: *Information Dashboard Design: The Effective Visual Communication of Data*. O’Reilly Media, Inc. 2006.
- [FGC⁺06] Farail, P.; GOUTILLET, P.; Canals, A.; Le Camus, C.; Sciamma, D.; Michel, P.; Crégut, X.: *The TOPCASED project: a toolkit in open source for critical aeronautic systems design*. *Ingenieurs de l’Automobile*. (781):54–59. 2006.
- [FGM⁺99] Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Masinter, L.; Leach, P.; Berners-Lee, T.: *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616 (Draft Standard). June 1999. Updated by RFCs 2817, 5785, 6266, 6585.
- [FGP⁺08] Ford, C.; Gileadi, I.; Purba, S.; Moerman, M.: *Patterns for Performance and Operability—Building and Testing Enterprise Software*. Auerbach Publications. Boston, MA, USA. 2008.
- [FHK09] Frank, U.; Heise, D.; Kattenstroth, H.: *Use of a Domain Specific Modeling Language for Realizing Versatile Dashboards*. *Proceedings of the 9th OOPSLA workshop on domain-specific modeling (DSM)*. 2009.
- [FHS⁺13] Fiedler, M.; Hauder, M.; Schneider, A.; Matthes, F.: *Foundations for the Integration of Enterprise Wikis and Specialized Tools for Enterprise Architecture Management*. In *11th International Conference on Wirtschaftsinformatik (WI), Leipzig, Germany*. 2013.
- [Fi54] Fitts, P. M.: *The information capacity of the human motor system in controlling the amplitude of movement*. *Journal of experimental psychology*. 47(6):381. 1954.

- [Fi13] Fink, A.: *How to conduct surveys: A step-by-step guide*. Sage. 5th edition. 2013.
- [FKF08] Fuchs-Kittowski, F.; Faust, D.: *The Semantic Architecture Tool (SemAT) for Collaborative Enterprise Architecture Development*. In (Briggs, R.; Antunes, P.; Vreede, G.-J.; Read, A., Ed.): *Groupware: Design, Implementation, and Use*. volume 5411 of *Lecture Notes in Computer Science*. pages 151–163. Springer Berlin Heidelberg. 2008.
- [FM11] Fette, I.; Melnikov, A.: *The WebSocket Protocol*. RFC 6455 (Proposed Standard). December 2011.
- [Fo97] Fong, J.: *Converting relational to object-oriented databases*. *ACM SIGMOD Record*. 26(1):53–58. 1997.
- [Fo99] Fowler, M.: *Refactoring: improving the design of existing code*. Addison-Wesley Professional. 1999.
- [FPB⁺12] Farwick, M.; Pasquazzo, W.; Breu, R.; Schweda, C. M.; Voges, K.; Hanschke, I.: *A Meta-Model for Automated Enterprise Architecture Model Maintenance*. In (Chi, C.-H.; Gasevic, D.; van den Heuvel, W.-J., Ed.): *EDOC*. pages 1–10. IEEE. 2012.
- [Fr06] Frank, U.: *Towards a pluralistic conception of research methods in information systems research*. Technical report. Universität Duisburg-Essen. 2006. ICB-Research Report.
- [Fr08] Frank, U.: *Integration — Reflections on a Pivotal Concept for Designing and Evaluating Information Systems*. In (Kaschek, R.; Kop, C.; Steinberger, C.; Fliedl, G., Ed.): *Information Systems and e-Business Technologies*. volume 5 of *Lecture Notes in Business Information Processing*. pages 111–122. Springer Berlin Heidelberg. 2008.
- [Fr10] Freeman, R. E.: *Strategic management: A stakeholder approach*. Cambridge University Press. 2010.
- [Fr11] Frank, U.: *Multi-Perspective Enterprise Modelling: Background and Terminological Foundation*. ICB-Research Report 46. Universität Duisburg-Essen. Essen. 2011.
- [Fr14] Freitag, A.: *Applying Business Capability Models in a Corporate Buyer M&A Process*. PhD thesis. Technical University Munich. 2014.
- [FSB⁺12] Farwick, M.; Schweda, C. M.; Breu, R.; Voges, K.; Hanschke, I.: *On Enterprise Architecture Change Events*. In *Trends in Enterprise Architecture Research and Practice-Driven Research on Enterprise Transformation*. pages 129–145. Springer Berlin Heidelberg. 2012.
- [FSB⁺14] Farwick, M.; Schweda, C.; Breu, R.; Hanschke, I.: *A Situational Method for Semi-automated Enterprise Architecture Documentation*. *Software & Systems Modeling (SOSYM)*. 2014.

- [FSS03] Falkovych, K.; Sabou, M.; Stuckenschmidt, H.: *UML for the Semantic Web: Transformation-Based Approaches*. In (Omelayenko, B.; Klein, M., Ed.): *Knowledge Transformation for the Semantic Web*. pages 92–106. IOS Press. 2003.
- [Ga71] Galbraith, J. R.: *Matrix organization designs How to combine functional and project forms*. *Business Horizons*. 14(1):29–40. 1971.
- [GHJ⁺94] Gamma, E.; Helm, R.; Johnson, R. E.; Vlissides, J.: *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley. 1994.
- [GKL⁺13] Gerth, C.; Küster, J. M.; Luckey, M.; Engels, G.: *Detection and resolution of conflicting change operations in version management of process models*. *Software & Systems Modeling*. 12(3):517–535. 2013.
- [GKP07] Gruschko, B.; Kolovos, D.; Paige, R.: *Towards synchronizing models with evolving metamodels*. In *Proceedings of the International Workshop on Model-Driven Software Evolution*. 2007.
- [GMR12] Grunow, S.; Matthes, F.; Roth, S.: *Towards Automated Enterprise Architecture Documentation: Data Quality Aspects of SAP PI*. In *16th East European Conference on Advances in Databases and Information Systems (ADBIS)*. pages 103–113. Pozna, Poland. 2012.
- [Gr64] Gross, B. M.: *The Managing of Organizations: The Administrative Struggle*. Free Press of Glencoe. 1964.
- [Gr91] Griswold, W. G.: *Program restructuring as an aid to software maintenance*. PhD thesis. University of Washington. 1991.
- [Gr12] Grunow, S.: *Automated Enterprise Service Bus based Enterprise Architecture-Documentation*. Master’s thesis. Technical University Munich. 2012.
- [GT95] Goodhue, D. L.; Thompson, R. L.: *Task-Technology Fit and Individual Performance*. *MIS Quarterly*. 19(2):213–236. 1995.
- [Ha05] Halpin, T. A.: *ORM 2*. In *On the Move to Meaningful Internet Systems (OTM) Workshops*. pages 676–687. Springer. 2005.
- [HA08] Heer, J.; Agrawala, M.: *Design considerations for collaborative visual analytics*. *Information Visualization*. 7(1):49–62. February 2008.
- [Ha10] Hanschke, I.: *Strategic IT Management – A Toolkit for Enterprise Architecture Management*. Springer. Berlin, Germany. 2010.
- [Ha13] Hauder, M.: *Bridging the Gap between Social Software and Business Process Management: A Research Agenda*. In *7th International Conference on Research Challenges in Information Science (RCIS), Paris, France*. 2013.
- [HBJ09] Herrmannsdoerfer, M.; Benz, S.; Juergens, E.: *COPE—automating coupled evolution of metamodels and models*. In *ECOOOP 2009—Object-Oriented Programming*. pages 52–76. Springer. 2009.

- [HBL⁺12] Holm, H.; Buschle, M.; Lagerström, R.; Ekstedt, M.: *Automatic data collection for enterprise architecture models. Software & Systems Modeling.* pages 1–17. 2012.
- [HC93] Hammer, M.; Champy, J.: *Reengineering the Corporation.* Harper Business. 1993.
- [HCL05] Heer, J.; Card, S. K.; Landay, J. A.: *prefuse : A Toolkit for Interactive Information Visualization. Information Visualization.* pages 421–430. 2005.
- [He78] Heckel, P.: *A technique for isolating differences between files. Communications of the ACM.* 21(4):264–268. 1978.
- [He07] Hevner, A. R.: *The three cycle view of design science research. Scandinavian journal of information systems.* 19(2):87. 2007.
- [HG01] Hakimpour, F.; Geppert, A.: *Resolving semantic heterogeneity in schema integration. In Proceedings of the international conference on Formal Ontology in Information Systems (FOIS).* pages 297–308. New York, NY, USA. 2001. ACM.
- [Hi02] Hinrichs, H.: *Datenqualitätsmanagement in Data Warehouse-Systemen.* PhD thesis. Universität Oldenburg. 2002.
- [HK87] Hull, R.; King, R.: *Semantic database modeling: survey, applications, and research issues. ACM Computing Surveys (CSUR).* 19(3):201–260. 1987.
- [HM76] Hunt, J. W.; McIlroy, M. D.: *An algorithm for differential file comparison.* Technical Report 41. AT&T Bell Laboratories Inc. 1976.
- [HM79] Hammer, M.; McLeod, D.: *On Database Management System Architecture.* Technical Report 79-4. Computer Science Department, University of Southern California, Los Angeles, CA. 1979.
- [HM85] Heimbigner, D.; McLeod, D.: *A federated architecture for information management. ACM Transactions on Information Systems (TOIS).* 3(3):253–278. 1985.
- [HM10] Halpin, T.; Morgan, T.: *Information modeling and relational databases.* Morgan Kaufmann. 2010.
- [HMP⁺04] Hevner, A. R.; March, S. T.; Park, J.; Ram, S.: *Design Science in Information Systems Research. MIS Quarterly.* 28(1):75–105. 2004.
- [HMR12] Hauder, M.; Matthes, F.; Roth, S.: *Challenges for Automated Enterprise Architecture Documentation. In Trends in Enterprise Architecture Research and Practice-Driven Research on Enterprise Transformation.* pages 21–39. Springer. 2012.
- [HMR⁺12] Hauder, M.; Matthes, F.; Roth, S.; Schulz, C.: *Generating dynamic cross-organizational process visualizations through abstract view model pattern matching. In Architecture Modeling for Future Internet enabled Enterprise (AMFInE).* 2012.

-
- [Hö13] Höfler, T.: *Enabling realtime collaborative data-intensive Web Applications — A case study using server-side JavaScript*. Master's thesis. Technical University Munich. 2013.
- [HOT76] Hall, P. A. V.; Owlett, J.; Todd, S.: *Relations and Entities*. In *IFIP Working Conference on Modelling in Data Base Management Systems*. pages 201–220. 1976.
- [HRP⁺13a] Hauder, M.; Roth, S.; Pigat, S.; Matthes, F.: *A Configurator for Visual Analysis of Enterprise Architectures*. In *ACM/IEEE 16th International Conference on Model Driven Engineering Languages and Systems (MODELS 2013)*. Miami, FL, USA. 2013.
- [HRP⁺13b] Hauder, M.; Roth, S.; Pigat, S.; Matthes, F.: *Tool Support for Conflict Resolution of Models for Automated Enterprise Architecture Documentation*. In *ACM/IEEE 16th International Conference on Model Driven Engineering Languages and Systems (MODELS 2013)*. Miami, FL, USA. 2013.
- [HRS⁺14] Hauder, M.; Roth, S.; Schulz, C.; Matthes, F.: *Agile Enterprise Architecture Management: An Analysis on the Application of Agile Principles*. In *4th International Symposium on Business Modeling and Software Design (BMSD)*,. Grand Duchy of Luxembourg, Luxembourg. 2014.
- [HS77] Hunt, J. W.; Szymanski, T. G.: *A fast algorithm for computing longest common subsequences*. *Communications of the ACM*. 20(5):350–353. 1977.
- [HS08] Happel, H.-J.; Seedorf, S.: *Documenting Service-Oriented Architectures with Ontobrowse Semantic Wiki*. In (Heinzl, A.; Appelrath, H.-J.; Sinz, E. J., Ed.): *Proceedings of the PRIMMIUM Subconference at the Multikonferenz Wirtschaftsinformatik (MKWI)*. volume 328 of *CEUR Workshop Proceedings*. 2008.
- [HSR⁺13] Hauder, M.; Schulz, C.; Roth, S.; Matthes, F.: *Organizational Factors Influencing Enterprise Architecture Management Challenges*. In *21st European Conference on Information Systems (ECIS), Utrecht, Netherland*. 2013.
- [HT07] Hanson, R.; Tacy, A.: *GWT in Action: Easy Ajax with the Google Web Toolkit*. Dreamtech Press. 2007.
- [HVT98] Hunt, J. J.; Vo, K.-P.; Tichy, W. F.: *Delta algorithms: An empirical analysis*. *ACM Transactions on Software Engineering and Methodology (TOSEM)*. 7(2):192–214. 1998.
- [IC95] Inkpen, A.; Choudhury, N.: *The seeking of strategy where it is not: Towards a theory of strategy absence*. *Strategic Management Journal*. 16(4):313–323. 1995.
- [ID10] Indurkha, N.; Damerau, F. J., Ed. *Handbook of natural language processing*. CRC Press. 2nd edition. 2010.

- [IE00] IEEE: *IEEE Std 1471-2000 for Recommended Practice for Architectural Description of Software-Intensive Systems*. 2000.
- [In97] ISO: *ISO/IEC 14772-1:1997 The Virtual Reality Modeling Language*. 1997.
- [In07] International Organization for Standardization: *ISO/IEC 42010:2007 Systems and software engineering – Recommended practice for architectural description of software-intensive systems*. 2007.
- [ISO04] ISO: *ISO/IEC 11179-1: Information technology — Metadata registries (MDR). Part 1: Framework*. 2004.
- [ISO07a] International Organization of Standardization: *ISO/IEC 11404:2007 Information technology – General-Purpose Datatypes (GPD)*. 2007.
- [ISO07b] International Organization for Standardization: *ISO/IEC 42010:2007 Systems and software engineering – Recommended practice for architectural description of software-intensive systems*. 2007.
- [JAB⁺08] Jouault, F.; Allilaire, F.; Bézivin, J.; Kurtev, I.: *ATL: A model transformation tool*. *Science of Computer Programming*. 72(1–2):31 – 39. 2008. Special Issue on Second issue of experimental software and toolkits (EST).
- [JG14] JGraph: *JavaScript HTML5 Diagramming Library Component*. 2014. . Online <http://www.jgraph.com/> (last accessed: Saturday 1st August, 2015).
- [JK06] Jouault, F.; Kurtev, I.: *Transforming models with ATL*. In *Satellite Events at the MoDELS 2005 Conference*. pages 128–138. Springer Berlin Heidelberg. 2006.
- [JLD⁺06] Jonkers, H.; Lankhorst, M. M.; ter Doest, H. W.; Arbab, F.; Bosma, H.; Wieringa, R. J.: *Enterprise architecture: Management tool and blueprint for the organisation*. *Information Systems Frontiers*. 8(2):63–66. 2006.
- [Jo14] Joyent, I.: *node.js*. 2014. . Online <http://nodejs.org/> (last accessed: Saturday 1st August, 2015).
- [KAV05] Kaisler, S.; Armour, F.; Valivullah, M.: *Enterprise Architecting: Critical Problems*. In *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on*. pages 224b–224b. 2005.
- [KC86a] Khoshafian, S. N.; Copeland, G. P.: *Object Identity*. In *Conference Proceedings on Object-oriented Programming Systems, Languages and Applications*. OOPLSA '86. pages 406–416. New York, NY, USA. 1986. ACM.
- [KC86b] Khoshafian, S. N.; Copeland, G. P.: *Object Identity*. *SIGPLAN Notices*. 21(11):406–416. June 1986.
- [KC04] Kimball, R.; Caserta, J.: *The data warehouse ETL toolkit*. John Wiley & Sons. 2004.

- [KDG13] Krause, C.; Dyck, J.; Giese, H.: *Metamodel-Specific Coupled Evolution Based on Dynamically Typed Graph Transformations*. In *Theory and Practice of Model Transformations, Proceedings of the 6th International Conference on Model Transformation (ICMT)*. volume 7909 of *LNCS*. pages 76–91. Springer. 2013.
- [KDRP⁺09] Kolovos, D. S.; Di Ruscio, D.; Pierantonio, A.; Paige, R. F.: *Different models for model matching: An analysis of approaches to support model differencing*. In *Comparison and Versioning of Software Models, 2009. CVSM'09. ICSE Workshop on*. pages 1–6. IEEE. 2009.
- [Ke97] Kent, S.: *Constraint Diagrams: Visualizing Invariants in Object-oriented Models*. In *Proceedings of the 12th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications. OOPSLA '97*. pages 327–341. New York, NY, USA. 1997. ACM.
- [Ke12] Keller, W.: *IT-Unternehmensarchitektur: von der Geschäftsstrategie zur optimalen IT-Unterstützung*. dpunkt.verlag. Heidelberg, Germany. 2nd edition. 2012.
- [KFH13] Kattenstroth, H.; Frank, U.; Heise, D.: *Towards a Modelling Method in Support of Evaluating Information Systems Integration*. In *EMISA*. pages 85–99. 2013.
- [KGI⁺13] Karvounarakis, G.; Green, T. J.; Ives, Z. G.; Tannen, V.: *Collaborative data sharing via update exchange and provenance*. *ACM Transactions on Database Systems (TODS)*. 38(3):19. 2013.
- [KHL⁺10] Koegel, M.; Herrmannsdoerfer, M.; Li, Y.; Helming, J.; David, J.: *Comparing State- and Operation-Based Change Tracking on Models*. In *14th IEEE International Enterprise Distributed Object Computing (EDOC) Conference*. pages 163–172. Los Alamitos, CA, USA. 2010. IEEE Computer Society.
- [KHS09] Koegel, M.; Helming, J.; Seyboth, S.: *Operation-based conflict detection and resolution*. In *Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models. CVSM '09*. pages 43–48. Washington, DC, USA. 2009. IEEE Computer Society.
- [Ki12] Kirschner, B.: *Graphical Interaction on Enterprise Architecture Visualisations*. Bachelor's thesis. Technical University Munich. 2012.
- [Ki14] Kirschner, B.: *Tool Support for Federated EA Model Management — an Industrial Case Study*. Master's thesis. Technical University Munich. 2014.
- [KKK13] Kelter, U.; Kehrer, T.; Koch, D.: *Patchen von Modellen*. In *Software Engineering 2013*. 2013.
- [KKO⁺12] Kehrer, T.; Kelter, U.; Ohrndorf, M.; Sollbach, T.: *Understanding model evolution through semantically lifting model differences with SiLift*. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*. pages 638–641. sep 2012.

- [KKP⁺12] Kehrer, T.; Kelter, U.; Pietsch, P.; Schmidt, M.: *Adaptability of model comparison tools*. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. ASE 2012. pages 306–309. New York, NY, USA. 2012. ACM.
- [KKT11] Kehrer, T.; Kelter, U.; Taentzer, G.: *A rule-based approach to the semantic lifting of model differences in the context of model versioning*. In *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*. pages 163–172. IEEE Computer Society. 2011.
- [KLS95] Krogstie, J.; Lindland, O. I.; Sindre, G.: *Defining quality aspects for conceptual models*. *ISCO*. 1995:216–231. 1995.
- [KPP95] Kitchenham, B.; Pickard, L.; Pfleeger, S. L.: *Case studies for method and tool evaluation*. *Software, IEEE*. 12(4):52–62. 1995.
- [KPR13] Kelter, U.; Pietsch, P.; Ringert, J. O.: *Report on the International Workshop on Comparison and Versioning of Software Models (CVSM)*. *Softwaretechnik-Trends*. 33:81–83. 2013.
- [Kr95] Krogstie, J.: *Conceptual Modeling for Computerized Information Systems Support in Organizations*. PhD thesis. Norwegian Institute of Technology. 1995.
- [KR02] Kimball, R.; Ross, M.: *The data warehouse toolkit: the complete guide to dimensional modelling*. John Wiley & Sons. Second edition. 2002.
- [KR07] Kishore, R.; Ramesh, R.: *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*. Integrated Series in Information Systems. Springer. 2007.
- [Kr10] Krcmar, H.: *Informationsmanagement*. Springer. 5th edition. 2010.
- [KR14] Kirschner, B.; Roth, S.: *Federated Enterprise Architecture Model Management: Collaborative Model Merging for Repositories with Loosely Coupled Schema and Data*. In *MKWI*. 2014.
- [KS91] Kim, W.; Seo, J.: *Classifying schematic and data heterogeneity in multidatabase systems*. *Computer*. 24(12):12–18. 1991.
- [KS03] Kalfoglou, Y.; Schorlemmer, M.: *Ontology mapping: the state of the art*. *The knowledge engineering review*. 18(1):1–31. 2003.
- [KT08] Kelly, S.; Tolvanen, J.-P.: *Domain-specific modeling: enabling full code generation*. Wiley. 2008.
- [Kü06] Kühne, T.: *Matters of (meta-) modeling*. *Software & Systems Modeling*. 5(4):369–385. 2006.
- [KW07] Kurpjuweit, S.; Winter, R.: *Viewpoint-based meta model engineering*. In *Enterprise Modelling and Information Systems Architectures—Concepts and*

- Applications, Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA), St. Goar, Germany.* pages 143–161. 2007.
- [KWN05] Kelter, U.; Wehren, J.; Niere, J.: *A Generic Difference Algorithm for UML Models.* In *Software Engineering.* volume 64 of *Lectures Notes in Informatics (LNI).* pages 105–116. 2005.
- [La13] Lankhorst, M.: *Enterprise architecture at work: Modelling, communication and analysis.* Springer. third edition. 2013.
- [LBE⁺82] Litwin, W.; Boudenant, J.; Esculier, C.; Ferrier, A.; Glorieux, A.; La Chimia, J.; Kabbaj, K.: *SIRIUS System for Distributed Data Management.* In *Distributed Databases.* pages 311–366. 1982.
- [Le99] Lee, Y.: *Information modeling: From design to implementation.* In *Proceedings of the Second World Manufacturing Congress.* pages 315–321. 1999.
- [LEW⁺02] Loy, M.; Eckstein, R.; Wood, D.; Elliott, J.; Cole, B.: *Java swing.* O'Reilly Media, Inc. 2002.
- [LHB10] Lidwell, W.; Holden, K.; Butler, J.: *Universal Principles of Design: 125 Ways to Enhance Usability, Influence Perception, Increase Appeal, Make Better Design Decisions, and Tech Through Design [25 Additional Design Principles].* Rockport publishers. 2010.
- [LHS⁺06] Lopes, D.; Hammoudi, S.; de Souza, J.; Bontempo, A.: *Metamodel matching: Experiments and comparison.* In *International Conference on Software Engineering Advances.* pages 2–2. IEEE. 2006.
- [Li32] Likert, R.: *A technique for the measurement of attitudes.* *Archives of Psychology.* 22(140):1–55. 1932.
- [Li04] Lindholm, T.: *A three-way merge for XML documents.* In *Proceedings of the 2004 ACM symposium on Document engineering.* pages 1–10. ACM. 2004.
- [LKT06] Lindholm, T.; Kangasharju, J.; Tarkoma, S.: *Fast and simple XML tree differencing by sequence alignment.* In *ACM symposium on Document engineering.* pages 75–84. 2006.
- [LO92] Lippe, E.; van Oosterom, N.: *Operation-based merging.* In *Proceedings of the fifth ACM SIGSOFT symposium on Software development environments.* SDE 5. pages 78–87. New York, NY, USA. 1992. ACM.
- [LR82] Landers, T. A.; Rosenberg, R.: *An Overview of Multibase.* In *Distributed Databases.* pages 153–184. 1982.
- [LW13] Langer, P.; Wimmer, M.: *A Benchmark for Conflict Detection Components of Model Versioning Systems.* *Softwaretechnik-Trends.* 33(2). 2013.
- [Ma08] Matthes, F.: *Softwarekartographie.* *Informatik Spektrum.* 31(6):527–536. 2008.

- [MBL⁺08] Matthes, F.; Buckl, S.; Leitel, J.; Schweda, C. M.: *Enterprise Architecture Management Tool Survey 2008*. Technical report. Technical University Munich. 2008.
- [MC94] Malone, T. W.; Crowston, K.: *The interdisciplinary study of coordination*. *ACM Computing Surveys (CSUR)*. 26(1):87–119. 1994.
- [Mc06] McAfee, A. P.: *Enterprise 2.0: The dawn of emergent collaboration*. *MIT Sloan management review*. 47(3):21–28. 2006.
- [MCH03] Malone, T. W.; Crowston, K.; Herman, G. A.: *Organizing business knowledge: The MIT process handbook*. MIT press. 2003.
- [MD94] Munson, J. P.; Dewan, P.: *A flexible object merging framework*. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. pages 231–242. ACM. 1994.
- [ME84] Mannino, M. V.; Effelsberg, W.: *Matching techniques in global schema design*. In *Proceedings of the First International Conference on Data Engineering*. pages 418–425. IEEE Computer Society. 1984.
- [Me99] Mens, T.: *A Formal Foundation for Object-Oriented Software Evolution*. PhD thesis. Vrije Universiteit Brussel. 1999.
- [Me02] Mens, T.: *A state-of-the-art survey on software merging*. *IEEE Transactions on Software Engineering*. 28(5):449–462. 2002.
- [Me14] Mercurial community: *Mercurial SCM*. 2014. . Online <http://mercurial.selenic.com/> (last accessed: Saturday 1st August, 2015).
- [MG11] Mell, P.; Grance, T.: *The NIST Definition of Cloud Computing*. Technical Report 800-145. National Institute of Standards and Technology (NIST). Gaithersburg, MD. September 2011.
- [MH80] McLeod, D.; Heimbigner, D.: *A federated architecture for database systems*. In *Proceedings of the national computer conference*. pages 283–289. ACM. 1980.
- [Mi14] Microsoft Corporation: *[MS-PPTX]: PowerPoint (.pptx) Extensions to the Office Open XML File Format*. 2014.
- [MJB⁺09] Moser, C.; Junginger, S.; Brückmann, M.; Schöne, K.-M.: *Some Process Patterns for Enterprise Architecture Management*. In *Software Engineering (Workshops) — Patterns in Enterprise Architecture Management (PEAM)*. pages 19–30. 2009.
- [MMC⁺01] Malhotra, A.; Majchrzak, A.; Carman, R.; Lott, V.: *Radical innovation without collocation: A case study at Boeing-Rocketdyne*. *MIS quarterly*. pages 229–249. 2001.
- [MMS⁺03] Maedche, A.; Motik, B.; Stojanovic, L.; Studer, R.; Volz, R.: *Ontologies for enterprise knowledge management*. *Intelligent Systems, IEEE*. 18(2):26–33. 2003.

-
- [MNS12] Matthes, F.; Neubert, C.; Steinhoff, A.: *Facilitating Structuring of Information for Business Users with Hybrid Wikis*. In *Lecture Notes of Communications in Computer and Information Science (CCIS)*. 2012.
- [Mo65] Moore, G. E.: *Cramming more components onto integrated circuits*. *Electronics*. 38(8). 1965.
- [Mo83] Morgan, G.: *Rethinking corporate strategy: A cybernetic perspective*. *Human Relations*. 36(4):345–360. 1983.
- [Mo09] Moody, D.: *The “physics” of notations: toward a scientific basis for constructing visual notations in software engineering*. *Software Engineering, IEEE Transactions on*. 35(6):756–779. 2009.
- [MRM⁺00] Majchrzak, A.; Rice, R. E.; Malhotra, A.; King, N.; Ba, S.: *Technology Adaptation: The Case of a Computer-Supported Inter-Organizational Virtual Team*. *MIS quarterly*. 24(4). 2000.
- [MS58] March, J. G.; Simon, H. A.: *Organizations*. Wiley. Oxford, England. 1958.
- [MS95] March, S. T.; Smith, G. F.: *Design and natural science research on information technology*. *Decision support systems*. 15(4):251–266. 1995.
- [MVG06] Mens, T.; Van Gorp, P.: *A Taxonomy of Model Transformation*. *Electronic Notes in Theoretical Computer Science*. 152:125–142. 2006. Proceedings of the International Workshop on Graph and Model Transformation (GraMoT 2005) Graph and Model Transformation 2005.
- [My86] Myers, E. W.: *An $O(ND)$ difference algorithm and its variations*. *Algorithmica*. 1(1-4):251–266. 1986.
- [My11] Mykhashchuk, M.: *Reverse Engineering of organization-specific Viewpoints: Applying and Extending Building Blocks for Enterprise Architecture Management Solutions (BEAMS)*. Master’s thesis. Technical University Munich. 2011.
- [MZ08] Malinowski, E.; Zimányi, E.: *Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications; with 10 Tables*. Springer. 2008.
- [Na04] National Information Standards Organization (NISO) Press: *Understanding metadata*. *National Information Standards*. 20. 2004.
- [Ne12] Neubert, C.: *Facilitating Emergent and Adaptive Information Structures in Enterprise 2.0 Platforms*. PhD thesis. Technical University Munich. München, Germany. 2012.
- [Ni05] Niemann, K. D.: *Von der Unternehmensarchitektur zur IT-Governance*. Vieweg. 2005.
- [NK04] Noy, N. F.; Klein, M.: *Ontology evolution: Not the same as schema evolution*. *Knowledge and information systems*. 6(4):428–440. 2004.

- [NMB⁺05] Nguyen, T. N.; Munson, E. V.; Boyland, J. T.; Thao, C.: *An infrastructure for development of object-oriented, multi-level configuration management services*. In *Proceedings of the 27th international conference on Software engineering*. pages 215–224. ACM. 2005.
- [No04] Noy, N. F.: *Semantic integration: a survey of ontology-based approaches*. *ACM Sigmod Record*. 33(4):65–70. 2004.
- [NP01] Niles, I.; Pease, A.: *Towards a standard upper ontology*. In *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*. pages 2–9. ACM. 2001.
- [NRS⁺99] Niswonger, L. H. R. M. B.; Roth, M. T.; Schwarz, P.; Wimmers, E.: *Transforming heterogeneous data with database middleware: Beyond integration*. *Data Engineering*. 22(1):32–38. 1999.
- [Ob10] Object Management Group (OMG): *Business Process Model and Notation (BPMN)*. 2010. . Online <http://www.omg.org/spec/BPMN/2.0/PDF> (last accessed: Saturday 1st August, 2015).
- [Ob13] Object Management Group (OMG): *Business Motivation Model (BMM)*. 2013. . Online <http://www.omg.org/spec/BMM/1.2/Beta2/PDF> (last accessed: Saturday 1st August, 2015).
- [ODa13] OASIS: *OData Version 4.0 Part 1: Protocol*. August 2013.
- [OK02] Ohst, D.; Kelter, U.: *A fine-grained version and configuration model in analysis and design*. In *Proceedings of the International Conference on Software Maintenance (ICSM)*. pages 521–527. 2002.
- [OMG11a] Object Management Group (OMG): *UML 2.4.1 Infrastructure Specification (formal/2011-08-05)*. 2011.
- [OMG11b] Object Management Group (OMG): *UML 2.4.1 Superstructure Specification (formal/2011-08-05)*. 2011.
- [Op92] Opdyke, W. F.: *Refactoring object-oriented frameworks*. PhD thesis. University of Illinois. 1992.
- [OPW⁺09] Op ’t Land, M.; Proper, E.; Waage, M.; Cloo, J.; Steghuis, C.: *Enterprise architecture: creating value by informed governance*. Springer. 2009.
- [Or93] Orlikowski, W. J.: *Case tools as organizational change: investigating incremental and radical changes in systems development*. *MIS quarterly*. 17(3). 1993.
- [Or02] Orlikowski, W. J.: *Knowing in practice: Enacting a collective capability in distributed organizing*. *Organization science*. 13(3):249–273. 2002.
- [Or13] Orhan, N.: *Visualization of Enterprise Architecture Model Evolution Based on an Example in the Consumer Goods Industry*. Master’s thesis. Technical University Munich. 2013.

-
- [Ou84] Ousterhout, J.: *Corner Stitching: A Data-Structuring Technique for VLSI Layout Tools*. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*. 3(1):87–100. January 1984.
- [OWK03] Ohst, D.; Welle, M.; Kelter, U.: *Differences between versions of UML diagrams*. *ACM SIGSOFT Software Engineering Notes*. 28(5):227–236. 2003.
- [Pa02] Patton, M. Q.: *Qualitative evaluation and research methods*. Sage Publications. 3 edition. 2002.
- [Pa14] Parr, T.: *StringTemplate*. 2014. . Online <http://www.stringtemplate.org/> (last accessed: Saturday 1st August, 2015).
- [PC05] Popfinger, C.; Conrad, S.: *Maintaining global integrity in federated relational databases using interactive component systems*. In *On the Move to Meaningful Internet Systems: CoopIS, DOA, and ODBASE*. pages 539–556. Springer. 2005.
- [Pe00] Pearson, K.: *On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling*. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*. 50(302):157–175. 1900.
- [PH90] Prahalad, C. K.; Hamel, G.: *The core competence of the corporation*. *Harvard business review*. 68(3):79–91. 1990.
- [Pi02] Pierce, B. C.: *Types and programming languages*. MIT press. 2002.
- [PLW⁺12] Ponta, S.; Laube, A.; van der Werf, J. M.; Gagnerot, G.; Blösch, M.; Verbeek, E.; van der Aalst, W.: *Policy and Security Configuration Management (PoSecCo): concept and architecture for automated model creation, population, maintenance and audit*. Technical report. 2012.
- [PMC⁺01] Purchase, H. C.; McGill, M.; Colpoys, L.; Carrington, D.: *Graph drawing aesthetics and the comprehension of UML class diagrams: an empirical study*. In (Eades, P.; Pattison, T., Ed.): *Australian Symposium on Information Visualisation*. Sydney, Australien. 2001.
- [Po80] Porter, M. E.: *Competitive strategies: Techniques for analyzing industries and competitors*. *The Free Press*. 1980.
- [Po07] Popfinger, C.: *Enhanced Active Databases for Federated Information Systems*. PhD thesis. Heinrich-Heine-Universität Düsseldorf. 2007.
- [PSV01] Perry, D. E.; Siy, H. P.; Votta, L. G.: *Parallel changes in large-scale software development: an observational case study*. *ACM Transactions on Software Engineering and Methodology (TOSEM)*. 10(3):308–337. 2001.
- [PW04] Peters, T. J.; Waterman, R. H.: *In search of excellence: Lessons from America's best-run companies*. Profile Books. Second edition. 2004.

- [Py05] Pyöriä, P.: *The concept of knowledge work revisited*. *Journal of Knowledge Management*. 9(3):116–127. 2005.
- [QHCW05] Quan-Haase, A.; Cothrel, J.; Wellman, B.: *Instant Messaging for Collaboration: A Case Study of a High-Tech Firm*. *Journal of Computer-Mediated Communication*. 10(4):00–00. 2005.
- [RB01a] Rahm, E.; Bernstein, P. A.: *On matching schemas automatically*. *VLDB Journal*. 10(4):334–350. 2001.
- [RB01b] Rahm, E.; Bernstein, P. A.: *A survey of approaches to automatic schema matching*. *VLDB Journal*. 10(4):334–350. 2001.
- [RC13] Rubin, J.; Chechik, M.: *N-way Model Merging*. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*. ESEC/FSE 2013. pages 301–311. New York, NY, USA. 2013. ACM.
- [RD00] Rahm, E.; Do, H. H.: *Data cleaning: Problems and current approaches*. *IEEE Data Engineering Bulletin*. 23(4):3–13. 2000.
- [Re13] Reschenhofer, T.: *Design and prototypical implementation of a model-based structure for the definition and calculation of Enterprise Architecture Key Performance Indicators*. Master’s thesis. Technical University Munich. 2013.
- [Re14] Redbooth: *Redbooth | The Most Complete Collaboration and Task Management Platform*. 2014. . Online <http://redbooth.com> (last accessed: Saturday 1st August, 2015).
- [RFG⁺05] Reddy, R.; France, R.; Ghosh, S.; Fleurey, F.; Baudry, B.: *Model composition-a signature-based approach*. In *Aspect Oriented Modeling (AOM) Workshop*. 2005.
- [RG03] Ramakrishnan, R.; Gehrke, J.: *Database management systems*. Osborne/McGraw-Hill. 3rd edition. 2003.
- [RH09] Runeson, P.; Höst, M.: *Guidelines for conducting and reporting case study research in software engineering*. *Empirical Software Engineering*. 14(2):131–164. 2009.
- [RHF⁺13] Roth, S.; Hauder, M.; Farwick, M.; Breu, R.; Matthes, F.: *Enterprise Architecture Documentation: Current Practices and Future Directions*. In *11th International Conference on Wirtschaftsinformatik (WI), Leipzig, Germany*. 2013.
- [RHM13a] Roth, S.; Hauder, M.; Matthes, F.: *Collaborative Evolution of Enterprise Architecture Models*. In *8th International Workshop on Models at Runtime (Models@run.time)*. Miami, USA. 2013.
- [RHM13b] Roth, S.; Hauder, M.; Matthes, F.: *Facilitating Conflict Resolution of Models for Automated Enterprise Architecture Documentation*. In *Nineteenth Americas Conference on Information Systems (AMCIS)*. 2013.

- [RHW⁺10] Rose, L. M.; Herrmannsdoerfer, M.; Williams, J. R.; Kolovos, D. S.; Garcés, K.; Paige, R. F.; Polack, F. A.: *A comparison of model migration tools*. In *Model Driven Engineering Languages and Systems*. pages 61–75. Springer. 2010.
- [RHZ⁺13] Roth, S.; Hauder, M.; Zec, M.; Utz, A.; Matthes, F.: *Empowering Business Users to Analyze Enterprise Architectures: Structural Model Matching to Configure Visualizations*. In *7th Workshop on Trends in Enterprise Architecture Research (TEAR 2013)*. Vancouver, Canada. 2013.
- [RL07] Robbes, R.; Lanza, M.: *A change-based approach to software evolution*. *Electronic Notes in Theoretical Computer Science*. 166:93–109. 2007.
- [RM13] Roth, S.; Matthes, F.: *Future Research Topics in Enterprise Architectures Evolution Analysis*. In *Software Engineering (SE) – Design for Future (DFF) Workshop*. 2013.
- [RM14] Roth, S.; Matthes, F.: *Visualizing Differences of Enterprise Architecture Models*. In *International Workshop on Comparison and Versioning of Software Models (CVSM) at Software Engineering (SE)*. Kiel, Germany. 2014.
- [Ro70] Royce, W. W.: *Managing the development of large software systems*. In *Proceedings of IEEE WESCON*. pages 1–11. 1970.
- [RRL⁺09] Rutle, A.; Rossini, A.; Lamo, Y.; Wolter, U.: *A Category-Theoretical Approach to the Formalisation of Version Control in MDE*. In (Chechik, M.; Wirsing, M., Ed.): *Fundamental Approaches to Software Engineering (FASE)*. volume 5503 of *Lecture Notes in Computer Science*. pages 64–78. Springer Berlin Heidelberg. 2009.
- [RSS⁺11] Reinhardt, W.; Schmidt, B.; Sloep, P.; Drachsler, H.: *Knowledge worker roles and actions—results of two empirical studies*. *Knowledge and Process Management*. 18(3):150–174. 2011.
- [RSV08] van der Raadt, B.; Schouten, S.; van Vliet, H.: *Stakeholder Perception of Enterprise Architecture*. In (Morrison, R.; Balasubramaniam, D.; Falkner, K., Ed.): *Software Architecture*. volume 5292 of *Lecture Notes in Computer Science*. pages 19–34. Springer Berlin Heidelberg. 2008.
- [RV08] Rivera, J. E.; Vallecillo, A.: *Representing and operating with model differences*. In *Objects, Components, Models and Patterns*. pages 141–160. Springer. 2008.
- [RW98] Rho, J.; Wu, C.: *An efficient version model of software diagrams*. In *Proceedings of the Fifth Asia Pacific Conference on Software Engineering*. pages 236–243. 1998.
- [RWR06] Ross, J. W.; Weill, P.; Robertson, D.: *Enterprise architecture as strategy: Creating a foundation for business execution*. Harvard Business Press. 2006.
- [RZM14] Roth, S.; Zec, M.; Matthes, F.: *Enterprise Architecture Visualization Tool Survey 2014*. Technical report. Technical University Munich. 2014.

- [Sa74] Saltzer, J. H.: *Protection and the control of information sharing in Multics. Communications of the ACM.* 17(7):388–402. 1974.
- [SBP⁺09] Steinberg, D.; Budinsky, F.; Paternostro, M.; Merks, E.: *EMF – Eclipse Modeling Framework (2nd edition)*. Addison-Wesley Longman. Amsterdam. 2009.
- [Sc97] Schaller, R. R.: *Moore’s law: past, present and future. Spectrum, IEEE.* 34(6):52–59. 1997.
- [Sc05] Schekkerman, J.: *The economic benefits of enterprise architecture*. Trafford Publishing. 2005.
- [Sc08] Schönherr, M.: *Towards a common terminology in the discipline of Enterprise Architecture*. In (Aier, S.; Johnson, P.; Schelp, J., Ed.): *Pre-Proceedings of the 3rd Workshop on Trends in Enterprise Architecture Research*. pages 107–123. Sydney, Australia. 2008.
- [Sc09] Schönherr, M.: *Towards a Common Terminology in the Discipline of Enterprise Architecture*. In (Feuerlicht, G.; Lamersdorf, W., Ed.): *Service-Oriented Computing – ICSOC 2008 Workshops*. pages 400–413. Berlin, Heidelberg, Germany. 2009. Springer.
- [Sc11] Schweda, C. M.: *Development of Organization-Specific Enterprise Architecture Modeling Languages Using Building Blocks*. PhD thesis. Technical University Munich. 2011.
- [Sc12] Schulz, C.: *A detailed process model for large scale data migration projects*. PhD thesis. Technical University Munich. 2012.
- [Sc13] Schrade, T.: *A Visual Tool for Conflict Resolution in EA Repositories*. Bachelor’s thesis. Technical University Munich. 2013.
- [SCT01] Shull, F.; Carver, J.; Travassos, G. H.: *An empirical methodology for introducing software processes. SIGSOFT Softw. Eng. Notes.* 26(5):288–296. September 2001.
- [SD92] Shen, H.; Dewan, P.: *Access control for collaborative environments*. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*. pages 51–58. ACM. 1992.
- [SE08] Shvaiko, P.; Euzenat, J.: *Ten challenges for ontology matching*. In *International Conference on Ontologies, Databases, and Applications of Semantics (ODBASE)*. pages 1163–1181. Springer. 2008.
- [SE13] Shvaiko, P.; Euzenat, J.: *Ontology matching: state of the art and future challenges. IEEE Transactions on Knowledge and Data Engineering.* 25:158–176. 2013.
- [SG89] Star, S. L.; Griesemer, J. R.: *Institutional ecology, ‘translations’ and boundary objects: Amateurs and professionals in Berkeley’s Museum of Vertebrate Zoology, 1907-39. Social studies of science.* 19(3):387–420. 1989.

-
- [SGT⁺11] Sousa, P.; Gabriel, R.; Tadao, G.; Carvalho, R.; Sousa, P. M.; Sampaio, A.: *Enterprise Transformation: The Serasa Experian Case*. In *Practice-Driven Research on Enterprise Transformation*. pages 134–145. Springer. 2011.
- [Si59] Simon, H. A.: *Theories of decision-making in economics and behavioral science*. *The American economic review*. 49(3):253–283. 1959.
- [Si96] Simon, H. A.: *The Sciences of the Artificial*. MIT Press. Cambridge, Massachusetts, USA. 3rd edition. 1996.
- [SK93] Sheth, A.; Kashyap, V.: *So far (schematically) yet so near (semantically)*. In *Proceedings of the IFIP WG2.6 Database Semantics Conference on Interoperable Database Systems*. pages 283–312. Lorne, Victoria, Australia. 1993.
- [SK07] Selonen, P.; Kettunen, M.: *Metamodel-based inference of inter-model correspondence*. In *11th European Conference on Software Maintenance and Reengineering (CSMR)*. pages 71–80. IEEE. 2007.
- [SL90] Sheth, A. P.; Larson, J. A.: *Federated database systems for managing distributed, heterogeneous, and autonomous databases*. *ACM Computing Surveys (CSUR)*. 22(3):183–236. 1990.
- [SLM⁺96] Steyaert, P.; Lucas, C.; Mens, K.; D’Hondt, T.: *Reuse contracts: Managing the evolution of reusable assets*. In *ACM Sigplan Notices*. volume 31. pages 268–285. ACM. 1996.
- [SMJ02] Spyns, P.; Meersman, R.; Jarrar, M.: *Data modelling versus ontology engineering*. *ACM SIGMod Record*. 31(4):12–17. 2002.
- [SMR12] Schaub, M.; Matthes, F.; Roth, S.: *Towards a Conceptual Framework for Interactive Enterprise Architecture Management Visualizations*. In *Modellierung*. Bamberg, Germany. 2012.
- [So11] Sommerville, I.: *Software engineering*. Pearson Higher Education. 9th edition. 2011.
- [SOB⁺08] Schmidt, A.; Osl, P.; Back, A.; Brenner, W.; Österle, H.: *A Method for Establishing Transparency on Integration Objects*. Technical Report BE HSG/CC CDQ/ 2. University of St. Gallen. 2008.
- [SP94] Spaccapietra, S.; Parent, C.: *View integration: A step forward in solving structural conflicts*. *Knowledge and Data Engineering, IEEE Transactions on*. 6(2):258–274. 1994.
- [Sp01] Spence, R.: *Information visualization*. ACM Press books. Addison-Wesley. Harlow, England. 2001.
- [SPD92] Spaccapietra, S.; Parent, C.; Dupont, Y.: *Model independent assertions for integration of heterogeneous schemas*. *VLDB Journal*. 1(1):81–126. 1992.

- [SPS11] Swenson, K. D.; Palmer, N.; Silver, B.: *Taming the Unpredictable: Real World Adaptive Case Management: Case Studies and Practical Guidance*. Future Strategies Inc. 2011.
- [SS75] Saltzer, J. H.; Schroeder, M. D.: *The protection of information in computer systems*. *Proceedings of the IEEE*. 63(9):1278–1308. 1975.
- [SS09] Staab, S.; Studer, R.: *Handbook on ontologies*. Springer. Second edition. 2009.
- [St73] Stachowiak, H.: *Allgemeine Modelltheorie*. Springer-Verlag. Wien, Austria. 1973.
- [St13] Steinhoff, A.: *Enhancing Tagging Systems with a Flexible, Faceted Organization Structure*. PhD thesis. Technical University Munich. München, Germany. 2013.
- [Su01] Sujansky, W.: *Heterogeneous Database Integration in Biomedicine*. *Journal of Biomedical Informatics*. 34(4):285–298. 2001.
- [SVV99] Speier, C.; Valacich, J. S.; Vessey, I.: *The influence of task interruption on individual decision making: An information overload perspective*. *Decision Sciences*. 30(2):337–360. 1999.
- [SW09] Schelp, J.; Winter, R.: *Language communities in enterprise architecture research*. In *DESRIST '09: Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*. New York, NY, USA. 2009. ACM.
- [Sw10] Swenson, K. D.: *Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done*. Meghan-Kiffer Press. 2010.
- [TBU10] Terwilliger, J. F.; Bernstein, P. A.; Unnithan, A.: *Automated Co-evolution of Conceptual Models, Physical Databases, and Mappings*. In *Proceedings of the 29th International Conference on Conceptual Modeling*. ER'10. pages 146–159. Berlin, Heidelberg. 2010. Springer-Verlag.
- [TBW⁺07] Treude, C.; Berlik, S.; Wenzel, S.; Kelter, U.: *Difference computation of large models*. In *Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering*. ESEC-FSE '07. pages 295–304. New York, NY, USA. 2007. ACM.
- [Te14] TeamViewer GmbH: *TeamViewer — Free Remote Control, Remote Access & Online Meetings*. 2014. . Online <http://www.teamviewer.com> (last accessed: Saturday 1st August, 2015).
- [TEF13] The Eclipse Foundation: *EMF Compare*. 2013. . Online <http://www.eclipse.org/emf/compare/> (last accessed: Saturday 1st August, 2015).

-
- [TEF14] The Eclipse Foundation: *Eclipse Modeling Framework Project (EMF)*. 2014. . Online <http://www.eclipse.org/emf/> (last accessed: Saturday 1st August, 2015).
- [TEL⁺10] Taentzer, G.; Ermel, C.; Langer, P.; Wimmer, M.: *Conflict detection for model versioning based on graph modifications*. In *Proceedings of the 5th international conference on Graph transformations*. ICGT'10. pages 171–186. Berlin, Heidelberg. 2010. Springer-Verlag.
- [TEL⁺12] Taentzer, G.; Ermel, C.; Langer, P.; Wimmer, M.: *A fundamental approach to model versioning based on graph modifications: from theory to implementation*. *Software & Systems Modeling*. 11:1–34. 2012.
- [TFN09] Tanner, A.; Feridun, M.; Nikulchenko, A.: *Fusio: semantic integration of systems management and enterprise information*. Technical report. IBM Technical Report RZ 3752. 2009.
- [Th05] Thomas, O.: *Das Modellverständnis in der Wirtschaftsinformatik: Historie, Literaturanalyse und Begriffsexplikation*. Technical report. Institut für Wirtschaftsinformatik (IW_i) – Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI). 2005.
- [Th11] The Open Group: *TOGAF Version 9.1*. Van Haren Publishing. 2011.
- [Th12a] The Open Group: *ArchiMate 2. 0 Specification*. Van Haren Publishing. 2012.
- [Th12b] The Open Group: *ArchiMate[®] 2.0: A Pocket Guide*. Van Haren Publishing. 2012.
- [Ti84] Tichy, W. F.: *The string-to-string correction problem with block moves*. *ACM Transactions on Computer Systems (TOCS)*. 2(4):309–321. 1984.
- [To70] Toffler, A.: *Future shock*. Random House Digital, Inc. 1970.
- [Tu01] Tufte, E. R.: *The Visual Display of Quantitative Information*. Graphics Press LLC. Ellicot City, MD, USA. 2nd edition. 2001.
- [TW93] Treacy, M.; Wiersema, F.: *Customer intimacy and other value disciplines*. *Harvard business review*. 71(1):84–93. 1993.
- [TW97] Treacy, M.; Wiersema, F. D.: *The discipline of market leaders: Choose your customers, narrow your focus, dominate your market*. Basic Books. 1997.
- [Va02] Vavouras, A.: *A metadata-driven approach for data warehouse refreshment*. PhD thesis. University of Zurich. 2002.
- [Va09] Vassiliadis, P.: *A Survey of Extract-Transform-Load Technology*. *International Journal of Data Warehousing & Mining*. 5(3):1–27. 2009.
- [Ve94] Venkatraman, N.: *IT-enabled business transformation: from automation to business scope redefinition*. *Sloan management review*. 35:73–73. 1994.

- [VGD99] Vavouras, A.; Gatziau, S.; Dittrich, K.: *The SIRIUS Approach for Refreshing Data Warehouses Incrementally*. In (Buchmann, A., Ed.): *Datenbanksysteme in Büro, Technik und Wissenschaft*. Informatik aktuell. pages 80–96. Springer. 1999.
- [VIR10] Voigt, K.; Ivanov, P.; Rummeler, A.: *MatchBox: Combined Meta-model Matching for Semi-automatic Mapping Generation*. In *Proceedings of the 2010 ACM Symposium on Applied Computing*. SAC '10. pages 2281–2288. New York, NY, USA. 2010. ACM.
- [Vo08] Vogels, W.: *Eventually Consistent*. *Queue*. 6(6):14–19. October 2008.
- [Vo09] Vogels, W.: *Eventually Consistent*. *Communications of the ACM*. 52(1):40–44. 2009.
- [VPZ88] Veijalainen, J.; Popescu-Zeletin, R.: *Multidatabase systems in ISO/OSI environment*. *Standards in Information Technology and Industrial Control, North-Holland, Netherlands*. pages 83–97. 1988.
- [W304a] W3C (W3C): *RDF Primer - W3C Recommendation*. February 2004. . Online <http://www.w3.org/TR/rdf-primer/> (last accessed: Saturday 1st August, 2015).
- [W304b] W3C: *XML Schema Part 2: Datatypes Second Edition*. 2004. . Online <http://www.w3.org/TR/xmlschema-2> (last accessed: Saturday 1st August, 2015).
- [W312] W3C (W3C): *OWL 2 Web Ontology Language Primer (Second Edition) - W3C Recommendation*. December 2012. . Online <http://www.w3.org/TR/owl2-primer/> (last accessed: Saturday 1st August, 2015).
- [Wa07] Wachsmuth, G.: *Metamodel adaptation and model co-adaptation*. In *ECOOP 2007-Object-Oriented Programming*. pages 600–624. Springer. 2007.
- [Wa12] Ware, C.: *Information visualization: perception for design*. Morgan Kaufmann Publishers. 3rd edition. 2012.
- [We98] Wenger, E.: *Communities of practice: Learning, meaning, and identity*. Cambridge university press. 1998.
- [We01] Wellman, B.: *Physical place and cyberplace: The rise of personalized networking*. *International journal of urban and regional research*. 25(2):227–252. 2001.
- [We08] Wenzel, S.: *Scalable visualization of model differences*. In *Proceedings of the 2008 international workshop on Comparison and versioning of software models*. CVSM '08. pages 41–46. New York, NY, USA. 2008. ACM.
- [We12] Weick, K. E.: *Making sense of the organization: Volume 2: The impermanent organization*. volume 2. John Wiley & Sons. 2012.

-
- [WF06] Winter, R.; Fischer, R.: *Essential Layers, Artifacts, and Dependencies of Enterprise Architecture*. In *Enterprise Distributed Object Computing Conference Workshops, 2006. EDOCW '06*. IEEE. 2006.
- [WF07] Winter, R.; Fischer, R.: *Essential Layers, Artifacts, and Dependencies of Enterprise Architecture*. *Journal of Enterprise Architecture*. 3(2):7–18. 2007.
- [Wi07] Wittenburg, A.: *Softwarekartographie: Modelle und Methoden zur systematischen Visualisierung von Anwendungslandschaften*. PhD thesis. Technical University Munich. 2007.
- [Wi11] Wieland, K.: *Conflict-tolerant Model Versioning*. PhD thesis. Vienna University of Technology. 2011.
- [WL13] Wimmer, M.; Langer, P.: *A Benchmark for Model Matching Systems: The Heterogeneous Metamodel Case*. *Softwaretechnik-Trends*. 33(2):101–104. 2013.
- [WLS⁺12] Wieland, K.; Langer, P.; Seidl, M.; Wimmer, M.; Kappel, G.: *Turning Conflicts into Collaboration - Concurrent Modeling in the Early Phases of Software Development*. *Computer Supported Cooperative Work: The Journal of Collaborative Computing*. tba:1–52. 2012.
- [WR09] Weill, P.; Ross, J.: *IT Savvy: What Top Executives Must Know to Go from Pain to Gain*. Harvard Business Press. 2009.
- [WRH⁺12] Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M. C.; Regnell, B.; Wesslén, A.: *Experimentation in software engineering*. Springer. 2012.
- [WW02] Webster, J.; Watson, R. T.: *Analyzing the Past to Prepare for the Future: Writing a Literature Review*. *MIS Quarterly*. 26(2):xiii—xxiii. 2002.
- [XS06] Xing, Z.; Stroulia, E.: *Refactoring detection based on UMLDiff change-facts queries*. In *13th Working Conference on Reverse Engineering (WCRE)*. pages 263–274. IEEE. 2006.
- [YCH03] Yang, C. C.; Chen, H.; Hong, K.: *Visualization of large category map for Internet browsing*. *Decision Support Systems*. 35(1):89–102. 2003.
- [Yi09] Yin, R.: *Case Study Research: Design and Methods*. Applied Social Research Methods. SAGE Publications. 4th edition. 2009.
- [YP13] Young, R.; Poon, S.: *Top management support—almost always necessary and sometimes sufficient for success: Findings from a fuzzy set analysis*. *International Journal of Project Management*. 31(7):943 – 957. 2013.
- [Za87] Zachman, J. A.: *A framework for information systems architecture*. *IBM System Journal*. 26(3):276–292. 1987.
- [Za97] Zachman, J. A.: *Enterprise architecture: The issue of the century*. *Database Programming and Design*. 10(3):44–53. 1997.

Bibliography

- [Zl77] Zloof, M. M.: *Query-by-example: A data base language*. *IBM systems Journal*. 16(4):324–343. 1977.
- [Zu97] Zukowski, J.: *Java AWT reference*. volume 3. O'Reilly Media, Inc. 1997.

List of Acronyms

ACID	Atomicity, Consistency, Isolation, Durability
ACL	Access Control List
ACM	Adaptive Case Management
AI	Artificial Intelligence
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ARS	Abstract Rewriting System
AST	Abstract Syntax Tree
ATL	ATLAS Transformation Language
AWT	Abstract Window Toolkit
BPM	Business Process Management
BPMN	Business Process Modeling Notation
CCMC	Cross-Community Model Consistency
CDM	Canonical or Common Data Model
CI	Configuration Item
CIC	Conflict Information Container
CMDB	configuration management database
COTS	commercial off-the-shelf
CPU	Central Processing Unit
CRM	Customer Relationship Management
CSV	Comma-Separated Values
CTO	Chief Technology Officer
DBA	Database Administrator
DBMS	database management system

List of Acronyms

DOM	Document Object Model
DSL	domain specific language
DSM	Domain-Specific Modeling
DVCS	distributed version control system
DWH	data warehouse
EA	Enterprise Architecture
EJB	Enterprise Java Beans
EMF	Eclipse Modeling Framework
EPC	Event-driven Process Chain
ERM	Entity Relationship Model
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
ETL	extract, transform, and load
FDBS	federated database system
GB	gigabyte
GDM	Generic Data Model
GPS	Global Positioning System
GWT	Google Web Toolkit
HG	Mercurial
HR	human resources
HTML	Hypertext Markup Language
IaaS	Infrastructure as a Service
IEEE	Institute of Electrical and Electronics Engineers
I/O	input/output
ISO	International Organization for Standardization
IT	Information Technology
ITIL	Information Technology Infrastructure Library
ITSM	IT Service Management
JSF	JavaServer Faces
JSON	JavaScript Object Notation
KPI	key performance indicator
LOI	Letter of Intent

LSE	large scale enterprise
LUT	lookup table
MDBS	multidatabase systems
MDE	model-driven engineering
MOF	Meta Object Facility
MVC	Model View Controller
MxL	model expression language
NDA	non-disclosure agreement
NLP	Natural Language Processing
ODP	OpenDocument Presentation
OID	origin identifier
OMG	Object Management Group
OO	object-oriented
ORM	Object-Role Modeling
OWL	Web Ontology Language
PaaS	Platform as a Service
PI	Process Integration
PNG	Portable Network Graphics
POJO	Plain Old Java Object
PPM	project portfolio management
PPT/PPTX	Microsoft Powerpoint Format
RACI	Responsible Accountable Consulted Informed
RAM	random access memory
RDB	Relational Database
RDBMS	Relational Database Management System
RDF	Resource Description Framework
REST	Representational State Transfer
RFC	Requests for Comments
ROI	Return on Investment
SaaS	Software as a Service
SCM	source code management
SQL	Structured Query Language
SVG	Scalable Vector Graphics
SVN	Subversion
TOGAF	The Open Group Architecture Framework

List of Acronyms

UI	user interface
UID	unique identifier
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VCS	version control system
VDSL	visual domain specific language
VLSI	very-large-scale integration
VM	virtual machine
XLS/XLSX	Microsoft Excel Spreadsheet Format
XML	Extensible Markup Language

Symbols

- \mathbb{Y} , 171, 175, 176, 191
 \mathcal{A} , 138, 165–167, 170, 171
 \mathcal{C} , 171, 175, 176
 \mathcal{C} , 167
 \mathcal{V} , 167
 δ , 175, 191, 192, 194
 \mathcal{D}^A , 139, 167, 170, 171
 \mathcal{D}^O , 139, 165, 167, 170, 171
 \mathcal{E} , 33, 78–83, 137, 145, 153, 165, 166, 173, 175, 176, 180–182, 194
 ι , 166
 \mathcal{L} , 141
 \mathcal{M} , 79–83, 138, 141, 144–146, 162–165, 174–176, 180, 181, 187
 \mathcal{O} , 138, 165, 167, 170
 \mathcal{R} , 180
 ρ , 137, 180
 \mathcal{S} , 141
 \mathcal{T} , 193
 \mathbb{T} , 138, 139
 \mathbb{U} , 79, 81, 180, 181
 χ^2 , 3
-
- A**
- abstract view model, [228](#)
abstract visualization model, [228](#)
ABSTRACTELEMENT, 226
abstraction, [27](#)
abstraction gap, [112](#), **180**
Abstraction Gap Resolver, [182](#), 182, 183, 185
ABSTRACTRENDERINGVISITOR, 238
ABSTRACTTRANSFORMINGVISITOR, 239
- ABSTRACTVIEWPOINT, 226, 237, 238, 240, 265, 276
ABSTRACTVIEWPOINTFACTORY, 240
ABSTRACTVISUALIZATIONOBJECTVISITOR, 237–239
ACTIVE, 255
ACTIVEINTERACTIONS, 233, 234
ACTIVITY, 114
ancestor, [33](#)
APPLICATION, 67, 255, 256, 258, 259
APPLICATION COMPONENT, 179, 180, 182–185
APPLYCHANGES, 235
ArchiMate 2.0, 17, 61, 179
ARCHITECTURAL DESCRIPTION, 20, 21
Architecture
– Federal System, [216](#)
– Federated System, [215](#)
ARCHITECTURE, 20
ARROWS, 232
ASSET, 255
ATTRIBUTE, XV, 66, 67, 78, 134–138, 140, 141, 145, 146, 162, 164–166, 168–173, 176–178, 187, 192, 197–199, 201, 202, 208, 209, 212, 218, 220, 221, 235, 255, 272, 277–279, 282, 292
ATTRIBUTEDEFINITION, XV, 66–68, 134–141, 146, 154, 155, 164, 167, 169–171, 176, 178, 181, 187, 191, 192, 199–201, 204, 208, 218, 220
ATTRIBUTEDEFINTION, 171
-
- B**
- baseline time, [169](#)

- BLUR, 233
 boundary model elements, [82](#), 82, **83**
 Branch, **145**
 BRAND, 258
 BUSINESS APPLICATION, 67, 79–81
-
- C**
-
- CHANGE, XIV, 169, 171–174, 176–178, 188–192, 194, 196, 235
 CHANGEOWNER, 243
 CHANGESET, XIV, 134, 140, 168–174, 176, 177, 189, 191
 changesets
 – potentially conflicting, **171**
 CHEVRON, 232
 CI, 255, 258
 CIRCLE, 232
 classifier, [78](#)
 CLICK, 234, 243, 244
 CLUSTER, 227
 CLUSTERDATABINDING, 227
 co-evolution, [123](#)
 Collaborative real-time conflict resolution, **207**
 COLOR, 230, 231
 COLORGRADIENT, XV, 230
 COLORTRANSITIONS, XV, 230
 common origin, [31](#)
 COMPENSATINGINTERACTION, 235, 242, 245, 248
 COMPOSITEINTERACTION, 239
 COMPOSITEINTERACTIONS, 233
 COMPOSITESYMBOL, 228, 230, 231, 233, 239
 COMPOSITEVISUALIZATIONOBJECT, 228, 233, 242
 COMPUTER SCIENCE STUDENT, 102
 CONCERNS, 20, 21
 conflict detection, [68](#)
 conflict management dashboard, [107](#)
 conflict resolution, [68](#)
 conflict resolution strategy
 – strict, 189, 190
 – tolerant, [189](#), 191
 conflict task, [58](#)
 CONSTRAINT, 136, 141
 CONSTRAINTVIOLATION, 141, 191, 192, 276
 CONTEXT, 114
 correct, [83](#)
 correctness, [83](#)
 COST, 79, 80
 CREATE, 173, 178
 create import metamodel, [148](#)
 CREATEATTRIBUTE, 235
 CREATEOBJECT, 235
 CREATEREference, 235
 cross-community model consistency, [85](#), **187**
-
- D**
-
- data, [28](#)
 DATABINDING, 227, 240, 241, 265, 276
 DATABINDINGFACTORY, 240
 DEFINITIONS, 162
 DELETE, 173, 178
 DELETEATTRIBUTE, 235
 DELETEOBJECT, 235
 DELETEREference, 235
 delta-backward, [172](#)
 DEVICE, 182, 184
 diff, [31](#), 31
 diff3, [31](#)
 DIFFERENCE, 162, 165
 DIFFERENCETEXT, 233
 differencing, [34](#), 66, 123
 DOUBLE CLICK, 234
 DRAG & DROP, 234
 DROPSHADOW, 233
 DUMMYDATABINDING, 227
-
- E**
-
- EA
 – as software intensive system of systems, 19
 – documentation
 – – automated, 49
 – experts, 12, 66, 73, 74, 204, 248, 251, 252, 259, 260, 273, 278, 280, 283, 284, 286, 289
 – information, 2, 6, 55, 187
 – information model, 25
 – management, III, 2–6, 9, 11, 15, 16, 18, 21–28, 40, 47–49, 54, 55, 57, 60, 61, 63, 65, 66, 69, 70, 72, 74, 77–81, 101, 109, 111, 112, 116, 118, 120, 121, 123, 125, 126, 132–134, 147, 149, 198, 201, 202, 225, 228, 251, 252, 254, 261, 262, 264, 265, 267, 268, 273, 281, 282, 284, 286, 292–294

-
- metamodel, 19, 24, 53, 141, 179, 185
 - model, III, 2–4, 6, 15, 16, 19, 24–27, 32, 35, 37–39, 48–50, 52, 54–58, 61, 66–69, 71, 72, 74, 75, 78, 83, 88–91, 93, 95–97, 101–104, 106–108, 110, 112, 118, 119, 123, 124, 128, 131–133, 135, 136, 141–148, 150, 152, 161, 172, 174, 179, 185, 186, 190, 198, 204, 206, 224, 225, 254, 256, 258, 261, 263, 266, 268, 269, 271, 273, 276, 278, 279, 281, 282, 285–288, 291–294
 - model maintenance, XVII, 2–4, 6, 9–11, 32, 34, 46, **49**, 49, 50, 53–61, 73, 77, 267, 280, 283, 284, 286, 287, 290, 293
 - modeling, 266
 - repository, 12, 47, 50, 52–59, 61, 66–68, 72, 73, 76–78, 84, 85, 88–91, 93, 96, 103, 104, 116, 118, 123, 132, 133, 147, 148, 150, 152, 167, 183, 185, 187, 213, 216, 254, 261, 267, 268, 271–274, 277–282
 - team, XIII, 24–26, 55, 56, 58, 66, 69–72, 90, 254, 258, 265–267, 270, 273, 283, 290
 - EA Management
 - interrelationship with other disciplines, 21
 - process, 23
 - – collect, 24
 - – communicate, 25
 - – explain, 25
 - – feedback, 25
 - – involve, 25
 - – model, 24
 - – motivate, 24
 - – support, 25
 - EA OBJECTS, 269
 - EATTRIBUTES, 155
 - Ecore, 154
 - EAttribute, 154
 - EClass, 154
 - EDataType, 155
 - EReference, 155
 - EFFECT, 233
 - ELLIPSIS, 232
 - ELLIPSISARC, 232
 - endogenous, 250
 - enterprise
 - transformation, 1
 - Enterprise Architecture, **1**
 - ENVIRONMENT, 20
 - EObject, 155
 - EOBJECT, 155
 - EREFERENCES, 155
 - exogenous, 250
 - export metamodel, **88**, 88, 89, 91–93, 147, 148
 - export model, **88**, 95, 97, 98, 117
 - EXTERNAL CUSTOMER, 269
 - external system, 46, 134, 135, 186, 290
-
- F**
-
- federal metamodel, 5
 - federal consistency, **84**, 111
 - federal model, 81
 - federal system, 66, 85, 89, 91, 92, 95–116, 118–125, 128, 132, 133, 151, 160
 - federated databases
 - five-level schema architecture, 42
 - federated EA model environment, XIII, XIV, 4–7, 11, 13, 63, 64, 66–70, 72, 75, **81**, 81, 83, 84, 88, 93, 97, 102, 104, 109, 117, 118, 122, 128, 131, 133, 142, 149, 151, 256, 268, 285–287
 - Federated EA Model Management, XIV, 6, 7, 9–13, 23, 63, 65, 68–74, 76, 77, 85, 86, 88, 101, 107, 112, 114, 116–118, 120, 121, 123, 124, 126, 128, 131–134, 141, 142, 144, 150–153, 155, 161, 174, 176, 213, 216, 218, 224, 225, 248, 249, 251, 252, 255, 268, 270–273, 279–289, 292, 293
 - Federated EA Modeling Community, 71, 121
 - federated EA modeling environment, *5*
 - federated models, 81
-
- G**
-
- global consistency, 85
 - GLOW, 233
 - GOALS, 114
 - GROUP, 255, 258
-
- H**
-
- HARVEYBALL, 232
 - HIDE, 235, 242
 - HIGHLIGHT, 235
 - HOVER, 234

- human tasks, [102](#), [105](#)
HYBRIDDATABINDING, [227](#)
HYPERLINK, [230](#)
-
- I**
-
- ID, [235](#)
identity, [68](#), [68](#)
– reconciliation within a federation, [93](#)
IMAGE, [232](#), [243](#), [244](#)
import metamodel, [88](#), [88](#), [88](#), [89](#), [91–93](#), [148](#)
import model, [88](#), [96–99](#), [101](#), [117](#), [144](#)
import models, [102](#), [103](#)
importing, [66](#)
in conflict, [137](#)
incremental changes, [97](#)
information, [17](#)
INFORMATION FLOW, [80](#), [81](#)
information source, [3](#)
information sources, [XIII](#), [4–7](#), [9](#), [11](#), [23](#), [45–47](#), [49](#), [53](#), [56](#), [59](#), [66–68](#), [71–76](#), [78](#), [84](#), [85](#), [88–93](#), [95–102](#), [104–109](#), [111–113](#), [115–118](#), [120](#), [122–125](#), [128](#), [131–133](#), [136](#), [143–154](#), [157–161](#), [167](#), [174](#), [176–179](#), [186](#), [213](#), [215](#), [216](#), [218](#), [224](#), [235](#), [253–255](#), [258](#), [263](#), [270–272](#), [274](#), [275](#), [281–283](#), [285–290](#), [293](#)
information system, [1](#), [3](#), [5](#), [7–9](#), [11](#), [13](#), [15](#), [22](#), [30](#), [45](#), [47–49](#), [52](#), [61](#), [66](#), [72](#), [74](#), [75](#), [84](#), [85](#), [88](#), [99](#), [104](#), [106](#), [116](#), [117](#), [119](#), [124](#), [128](#), [132](#), [133](#), [135](#), [147](#), [148](#), [151](#), [152](#), [154](#), [158](#), [177](#), [186](#), [216](#), [262](#), [272](#), [293](#), [294](#)
INFRASTRUCTURE ELEMENT, [183](#)
INFRASTRUCTURE INTERFACE, [179](#), [180](#), [184](#), [185](#)
instance mappings, [68](#)
instanceOf, [141](#), [142](#)
integrate information sources, [147](#)
integration, [146](#)
INTERACTION, [233–235](#), [239](#), [242](#), [245](#), [247](#), [248](#)
interactive conflict management dashboard, [198](#)
INTERACTIVEPLANARSYMBOLS, [237](#)
INTERNAL DEPARTMENTS, [269](#)
INTERNAL SERVICES, [255](#)
INTERVALSCALEDCOLOR, [231](#)
- Isolated model elements, [82](#)
IT SERVICES, [269](#)
-
- K**
-
- KEYPRESSINTERACTION, [234](#)
knowledge worker, [69](#)
knowledge workers, [69](#)
KPI, [25](#)
-
- L**
-
- LIBRARY VIEWPOINT, [21](#)
LINE, [232](#)
LINECONNECTION, [232](#)
LINESTYLES, [232](#)
local consistency, [64](#), [84](#)
lossless conflict resolution, [110](#)
-
- M**
-
- MAINTENANCE CYCLE, [67](#), [68](#)
mapping
– Ecore, [154](#)
– relational databases, [156](#)
master-slave, [274](#)
master-slave relationship, [153](#), [275](#), [283](#), [289](#)
meta modeling, [29](#), [38](#), [279](#), [292](#), [294](#)
meta-information, [116](#), [117](#), [146](#), [157](#), [174](#), [176](#), [188](#), [224](#)
meta-meta-metamodel, [30](#)
meta-metamodel, [30](#), [91](#), [142](#)
metamodel, [III](#), [XIII–XV](#), [2](#), [5–7](#), [11](#), [19](#), [19](#), [28](#), [28–30](#), [34](#), [35](#), [38–40](#), [42](#), [45](#), [49](#), [53–55](#), [60](#), [61](#), [64–68](#), [72](#), [74](#), [76](#), [77](#), [83](#), [84](#), [88–93](#), [101](#), [102](#), [109](#), [111](#), [112](#), [116–118](#), [120](#), [122–125](#), [131–136](#), [141](#), [142](#), [144](#), [146–148](#), [154–157](#), [160](#), [162–164](#), [167](#), [169](#), [172](#), [174](#), [178](#), [179](#), [181](#), [189–192](#), [198](#), [199](#), [207](#), [208](#), [210](#), [216](#), [218](#), [225](#), [253–255](#), [259](#), [260](#), [263–265](#), [268](#), [269](#), [279](#), [282–284](#), [286](#), [288](#), [289](#), [291–295](#), [340–342](#)
– common, [89](#)
metamodel mappings, [66](#)
MISSION, [19](#)
MODEL, [XV](#), [21](#), [136](#), [138](#), [218–220](#)
model, [28](#), [28](#)
model conflict, [105](#)
– abstraction gap, [XIV](#), [151](#), [179–185](#), [254](#), [258](#)

- model creator, [28](#)
 Model differencing, [162](#)
 model element, [78](#)
 MODELELEMENT, XIV, XVI, 134–138, 140, 144–146, 162, 164, 166, 168–174, 176–178, 182, 183, 186, 190–192, 194–197, 256, 260–262, 266, 278, 283, 288, 289, 291, 295, 303
 – state
 – – in conflict, [137](#)
 – – normal, [137](#)
 – – under consolidation, [137](#)
 MODELELEMENT, 196
 modeling communities, [64](#), 65
 modeling community, [82](#)
 – team, [65](#)
 modeling expert, [29](#), 133
 modeling experts, [66](#)
 MODELINTERACTION, 234, 235, 244, 247
 MODELS, XV, 21, 135, 145, 164, 221, 222
 MOUSEINTERACTION, 234
 MOVE, 178
-
- N**
-
- n-way merge, [32](#), [33](#), 33
 NATIVEINTERACTION, 234
 necessary degree of freedom, [84](#)
 new, [137](#)
 NODE, 179
 NODE DEVICE, 179, 185
 NODE DEVICES, 179, 183
 NODES, 179, 180
 NOMINALSCALEDCOLOR, 231
-
- O**
-
- OBJECT, XV, 78, 134–140, 145, 146, 162, 164–170, 172, 176–178, 187, 191, 192, 195, 198–204, 208, 209, 212, 220, 221, 235, 258, 263, 264, 272, 274, 276–279, 282, 292
 object graph, [228](#), [228](#), 228, 240
 OBJECTDEFINITION, XV, 134–141, 145, 146, 150, 154, 162, 164, 165, 167, 169–171, 176, 178, 181, 187, 192, 198–201, 204, 205, 208, 219, 220, 256, 264, 265, 268, 272, 274, 276–278, 282
 OPENINNEWWINDOW, 244
 OPERATING SYSTEM, 67, 79–81
 OPERATION, XVI, 172, 174, 178, 196, 304
 ORDINALSCALEDCOLOR, 231
 origin, [33](#), 93
-
- P**
-
- PARALLELINTERACTION, 233, 243
 PASSIVEINTERACTION, 234, 235, 244
 PERSON, 258
 PLANARSYMBOL, 230–233, 237
 PNGRENDERER, 230
 POINT, 232
 POLYGON, 232
 POLYLINE, 232
 POWERPOINTRENDERER, 230
 preview model, 103, 146
 PRINCIPAL, 218
 problem domain, [15](#)
 PRODUCT, 255, 256, 258
 PROGRESSBAR, 233
 PROJECTS, 80
 PROPAGATE TASKS, 235
 provenance, [33](#)
 publish model changes, [66](#)
 publish model snapshots, [66](#)
 purpose, [28](#)
-
- R**
-
- Raphaël, 228, 242
 RAPHAELRENDERER, 233–235, 244
 RATIONALE, 20
 real-world, [28](#)
 RECTANGLE, 232
 RELATIONSHIP, 199–202, 259
 RENDERER, 226, 230, 233, 234, 237, 241
 RENDERERFACTORY, 226
 RENDERINGFACTORY, 241
 RENTED, 255
 REVOKECHANGES, 235
 ROLE, XVI, 114, 134, 135, 137, 139, 174, 177, 190, 303
 role, [69](#), [69](#), 143
 – data owner, 55, 56, 58–60, 67, 72, 88–92, 96, 107, 112, 120, 126, 144, 147, 148, 150, 168, 183, 184, 186, 187, 261, 262, 264, 275
 – data steward, 72
 – decision maker, 72

-
- EA coordinator, 55, 71, 89, 90, 92, 103, 108, 113, 147, 148, 150, 179, 186, 187
 - EA stakeholder, 59, 70–72, 90, 96, 98, 103, 106–108, 110, 113, 115, 116, 118, 119, 126, 152, 179, 183, 184, 186, 187, 254, 262, 270–272, 281, 283, 284, 290
 - enterprise architect, 3, 65–68, 70–72, 101, 103, 104, 107, 108, 113, 116, 118, 119, 131, 168, 206, 222, 258–266, 268–280, 282–284, 289
 - modeling expert, 67, 68, 71, 88–92, 113, 148, 150, 179, 182, 186, 198, 292
 - repository manager, 58, 59, 71, 89–92, 96–98, 101, 103, 105, 107, 110–113, 115, 136, 144, 148, 150, 182, 183, 186, 187, 198, 282
-
- S**
-
- SCALEDCOLOR, 231
 - scene graph, [228](#)
 - schema, [29](#)
 - semantic zoom, [225](#), [225](#)
 - semi-automated imports, [59](#)
 - SEPARATOR, 230
 - SEQUENTIALINTERACTION, 233, 243
 - SERVER, 80, 81
 - SERVICE, 255, 258
 - SETATTRIBUTE, 235
 - SETOBJECT, 235
 - SETREFERENCE, 235
 - SETSTATE, 235
 - SETTASKOWNER, 235
 - SETTASKSTATE, 235, 243
 - SHOW, 235, 242, 243
 - single point of truth, 6
 - SIZEVISITOR, 238, 239, 241
 - SLAs, 80, 81
 - SOFTWARE PRODUCTS, 269
 - SOLIDCOLOR, 230, 231
 - solution domain, [49](#)
 - space, [218](#)
 - STAKEHOLDERS, 20, 21
 - Standards
 - IEEE 1471, 19
 - ISO 42010, 19
 - STATE, 134, 137–139, 295
 - state-based, [168](#)
 - strict, [136](#)
 - strict merge strategy, [111](#)
 - STUDENT, 102
 - SUBSYSTEMS, 255
 - SUPPLIER, 258
 - SUPPORT, 269
 - surrogate key, [46](#)
 - SYMBOL, 228, 230–235, 239, 242
 - symbolic model, [228](#)
 - SYSTEM, 19, 20
 - system of systems, [19](#)
-
- T**
-
- TASK, XIV, 134, 137–140, 176, 177, 244, 295
 - task, [105](#), 142, **143**
 - state
 - – ignored, [139](#), 139
 - – new, 139
 - – resolved, [139](#)
 - – reviewed, [139](#)
 - TASKS, 114, 137, 138, 140, 174, 177, 190, 288, 295
 - tasks, [142](#), 142
 - TEXT, 231, 232, 240, 246
 - three-way differencing, [31](#), 102
 - token model, 141
 - tolerant merge strategy, [111](#), 111
 - TOUCHINTERACTION, 235
 - TRANSFORMINGVISITOR, 241
 - transient, [140](#)
 - two-way differencing, [31](#), 102
 - TYPE, 136, 155
 - Type order, **193**
-
- U**
-
- universe of discourse, [27](#)
 - UPDATE, 173, 178, 235, 243–245, 247
 - UPDATES, 173
 - USE, 178
-
- V**
-
- VALUE, 136, 155, 171, 178, 201
 - version control system
 - Git, 31
 - HG, 31
 - SVN, 31
 - VIEW, 20, 21
 - view data model, [228](#)

view model, [227](#), [228](#), [240](#), [241](#)
VIEWPOINT, [20](#), [21](#)
VIEWPOINTFACTORY, [226](#)
visual update, [247](#)
visualization model, [227](#), [228](#), [241](#)
VISUALIZATIONOBJECT, [225](#), [226](#), [228](#), [230](#),
[233](#), [237](#), [239](#), [241](#)
VISUALIZATIONPROCESSOR, [240](#)

W

worklist, [105](#), [105](#)
workspace, [181](#), [181](#), [218](#)

