

Fabian Muff

Metamodeling for Extended Reality

OPEN ACCESS



Springer

Metamodeling for Extended Reality

Fabian Muff

Metamodeling for Extended Reality

 Springer

Fabian Muff
Digitalization and Information Systems
Group
University of Fribourg
Fribourg, Switzerland



ISBN 978-3-031-76761-6 ISBN 978-3-031-76762-3 (eBook)
<https://doi.org/10.1007/978-3-031-76762-3>

The open access publication of this book has been published with the support of the Swiss National Science Foundation.

© The Editor(s) (if applicable) and The Author(s) 2025. This book is an open access publication.

Open Access This book is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this book are included in the book's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the book's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

If disposing of this product, please recycle the paper.

Preface

Conceptual modeling and metamodeling are disciplines with a long history. The aim of conceptual modeling is to formally describe certain elements of the tangible and social environment to enhance comprehension and facilitate communication. It is a crucial part of documenting and understanding knowledge in industry and one big area in the business informatics discipline. The generic aim of metamodeling is to create a shared collection of items and connections that can be reused in multiple modeling methods.

The technology of extended reality, which encompasses augmented reality, virtual reality, and mixed reality, has gained significant traction in recent years in both research and industry. These technologies digitally enhance reality with virtual content to varying degrees, with the aim of integrating digital content into the real or virtual world, enabling interaction with virtual information and the real world. Technological advances have made extended reality devices more powerful and affordable. This opens up the possibility of using extended reality technology in various areas, including conceptual modeling and metamodeling.

The combination of extended reality and metamodeling could make model creation and application more intuitive and integrated into everyday work practices, making modeling accessible and feasible for non-experts and seamlessly integrating it into everyday tasks. By using extended reality, complex concepts and processes can be visualized and interacted with in a real-world context, enhancing understanding and application. For instance, it can aid in visualizing business processes in the real world based on conceptual models.

However, metamodeling and conceptual modeling have traditionally been limited to two-dimensional representations, mostly on computer screens. This limitation restricts their practical application in real-world extended reality scenarios, since the real world has three dimensions.

This book contains the core parts of my dissertation that has been completed in 2024. It explores the challenges of metamodeling in the context of extended reality and emphasizes the need for new concepts in metamodeling to effectively combine it with extended reality technologies. The central question of this work is how metamodeling can be used “in” and “for” extended reality.

The core themes of this book have been presented at international conferences in the fields of business informatics, conceptual modeling, and computer science. The research for this book was conducted during my employment in the Department of Informatics at the University of Fribourg. I am grateful to the individuals who provided invaluable support and guidance throughout the research process, offering insights, advice, and constructive feedback on the topic.

I would like to express my gratitude to Prof. Dr. Hans-Georg Fill for his invaluable support and insightful discussions on the subject of conceptual modeling and extended reality. Furthermore, I am grateful for the opportunity to participate in international conferences and projects, where I was able to present the concepts presented in this book and receive constructive feedback from professionals in the field. Finally, I want to thank all my colleagues in the Digitalization and Information Systems Group at the University of Fribourg for the insightful discussions on conceptual modeling and information technology that have taken place over the past few years.

Fribourg, Switzerland

Fabian Muff

Contents

- 1 Introduction** 1
 - 1.1 Background and Motivation 4
 - 1.2 Research Objectives and Questions 7
 - 1.3 Methodology 8
 - 1.4 Outline 10
 - 1.5 Research Contributions 11
- 2 State-of-the-Art and Related Work** 17
 - 2.1 Modeling 17
 - 2.1.1 Conceptual Modeling 19
 - 2.1.2 Enterprise Modeling 21
 - 2.1.3 Metamodeling 22
 - 2.2 Extended Reality 27
 - 2.2.1 Virtual Reality 27
 - 2.2.2 Augmented Reality 29
 - 2.2.3 Metaverse 35
 - 2.2.4 Positioning of Extended Reality, Virtual Reality
and Augmented Reality 36
 - 2.3 Literature Study of Pairing Conceptual Modeling with VR/AR 36
 - 2.3.1 Methodology of the Analysis 38
 - 2.3.2 Results of the Literature Study 43
 - 2.3.3 Result of Topic Refinement 46
 - 2.3.4 Discussion of Findings 51
 - 2.3.5 Related Studies 56
 - 2.3.6 Summarized Findings of Literature Study 56
 - 2.3.7 Additional Data of the Analysis 60
- 3 Derivation of Generic Requirements for Metamodeling for
Extended Reality** 61
 - 3.1 Morphological Schemes for Augmented Reality
and Enterprise Modeling 62
 - 3.1.1 Description of the Solution Space 63

3.1.2	Limiting the Solution Space.....	64
3.1.3	Exemplary Use Case for the Strategic Perspective.....	64
3.1.4	Exemplary Use Case for the Business Process Perspective...	68
3.1.5	Exemplary Use Case for the IT Perspective.....	71
3.2	Resulting Generic Requirements.....	74
4	Specific Requirements for Metamodeling for Extended Reality.....	77
4.1	Methodology for Requirements Derivation.....	77
4.2	Three-Dimensional Coordinate Mappings.....	78
4.2.1	Coordinate Systems.....	78
4.2.2	Position, Orientation and Coordinate Transformations.....	81
4.2.3	Requirements of Coordinate Mappings.....	85
4.3	Visualization of 3D Model Components.....	86
4.3.1	Translation of Two-Dimensional Notations.....	87
4.3.2	Parametric Modeling.....	88
4.3.3	Polygonal Modeling.....	88
4.3.4	Dynamic Behavior of Three-Dimensional Visualizations....	90
4.3.5	Three-Dimensional Data Formats.....	91
4.3.6	Requirements for the Visualization of 3D Model Components.....	92
4.4	Detection and Tracking of the Environment and Real-World Objects.....	93
4.4.1	Device and Object Tracking.....	94
4.4.2	Environment Tracking.....	97
4.4.3	Requirements for the Detection and Tracking.....	100
4.5	Context.....	101
4.5.1	Mapping to Real-World Objects and Semantic Reasoning ...	101
4.5.2	Requirements for Context.....	108
4.6	Interaction.....	108
4.6.1	Physical Keyboard-Based Interaction.....	109
4.6.2	2D Mouse-Based Interaction.....	109
4.6.3	3D Mouse-Based Interaction.....	110
4.6.4	Voice-Based Interaction.....	110
4.6.5	Gesture-Based Interaction.....	111
4.6.6	Collaborative Interaction.....	111
4.6.7	Requirements for Interaction.....	112
4.7	Aggregated Generic and Specific Requirements for Joining Metamodeling with Extended Reality.....	113
4.7.1	Requirements for the Meta ² -Model and Modeling Methods.....	113
4.7.2	Need for Knowledge-Based Virtual and Augmented Reality Approaches.....	118

5 ARWFMM: A Modeling Method as an Example for Knowledge-Based Virtual and Augmented Reality	121
5.1 Related Approaches for Conceptual Modeling and Model-Driven Engineering for AR	123
5.2 Related Development and Metamodeling Platforms	125
5.2.1 Development Platforms	126
5.2.2 Metamodeling Platforms	130
5.2.3 Implications for the Derivation of the Modeling Method	139
5.3 Derivation of the Visual Modeling Method	139
5.3.1 Methodology	140
5.3.2 Requirements	140
5.3.3 Language Specification	143
5.3.4 Modeling Procedure	146
5.3.5 Mechanisms and Algorithms	146
5.3.6 First Implementation and Execution on ADOxx	147
5.4 Evaluation of the Visual Modeling Method on ADOxx	147
5.4.1 Use Case for the Domain-Specific Visual Modeling Method	148
5.4.2 Comparative Evaluation of the First Prototype	150
5.4.3 Formal Evaluation of the Domain-Specific Modeling Language	152
5.5 Limitations Regarding Usability	158
6 M2AR: An Architecture for a 3D Enhanced Metamodeling Platform for Extended Reality	163
6.1 Structure Base	164
6.1.1 M2AR Meta ² -Model: Meta-Layer	164
6.1.2 M2AR Meta ² -Model: Instance-Layer	171
6.2 Access and Persistency Service	174
6.3 M2AR Modeling Client	175
6.3.1 Metamodeling Client	175
6.3.2 Instance Modeling Client	184
6.4 Proposed Conceptual Architecture for M2AR	186
6.4.1 Global Shared Datastructure Module	186
6.4.2 Database	187
6.4.3 API Module	187
6.4.4 Metamodeling Client Module	188
6.4.5 Instance Modeling Client Module	188
6.4.6 Collaboration Module	189
6.5 Implications for Implementation	189
7 Prototypical Realization of the M2AR Metamodeling Platform	191
7.1 Technology Stack	191
7.2 Database	192
7.3 Global Shared Datastructure Module	194
7.4 API Server Module	195

7.5	Instance Modeling Client: M2AR Modeler	197
7.5.1	Visual Main Components of the Client	197
7.5.2	Client Communication	199
7.5.3	Modeling States	200
7.5.4	VizRep Implementation	201
7.5.5	3D Interaction and Animation Loop	205
7.5.6	Hybrid ARWFMM Implementation	206
7.5.7	ARWFMM AR Engine	206
7.6	Metamodeling Client	210
7.7	Collaboration Server	210
7.8	Need for a First Evaluation	211
8	Evaluation of the M2AR Platform Prototype	213
8.1	Evaluation Against Requirements	213
8.2	Demonstration of the M2AR Implementation	217
8.2.1	ARWFMM Implementation on the M2AR Metamodeling Platform	217
8.2.2	Furniture Assembly Use Case	221
8.2.3	Machine Process Use Case	225
8.2.4	Office Tour Use Case	226
8.2.5	Discussion of Demonstration	229
8.3	Empirical Evaluation of ARWFMM Concepts on M2AR	230
8.3.1	Study Design	231
8.3.2	Experimental Subjects	231
8.3.3	Evaluation Metrics	234
8.3.4	Results	234
8.3.5	Threats to Validity	236
8.4	Summary of the M2AR Platform Prototype Evaluation	237
9	Summary and Outlook	239
9.1	Alignment with the Research Questions	239
9.2	Limitations and Future Research	240
9.3	Summary	241
A	ARWFMM Use Case Example with FDMM	243
	Bibliography	245

Acronyms

2D	two-dimensional
3D	three-dimensional
API	application programming interface
AR	augmented reality
AREA	Augmented Reality for Enterprise Alliance
ARWFMM	Augmented Reality Workflow Modeling Method
AV	augmented virtuality
BMC	business model canvas
CAD	computer-aided design
CTM	correlated topic model
DSL	domain-specific language
DSML	domain-specific modeling language
DSR	design science research
EM	enterprise modeling
ETM	embedding-based topic model
FCO	first class object
GLTF	Graphics Library Transmission Format
GME	generic modeling environment
GPS	Global Positioning System
HMD	head-mounted display
IIoT	Industrial Internet of Things
IoT	Internet of Things
IRR	inter-rater reliability
LDA	latent dirichlet allocation
MR	mixed reality
NMF	non-negative matrix factorization
OCL	object constraint language
OMG	Object Management Group
OWL	web ontology language
PwC	PricewaterhouseCoopers
QR	quick response

SATM	self-aggregation-based topic model
SDK	software development kit
SQL	structured query language
STM	structural topic model
TF-IDF	term frequency inverse document frequency
UI	user interface
UML	unified modeling language
USD	universal scene description
UUID	universally unique identifier
VR	virtual reality
XR	extended reality

Chapter 1

Introduction



Globalization has been a significant development in human history. It is characterized by the increasing interdependence of different parts of the world, leading to unprecedented changes in areas such as economics, culture, politics, technology, environment, society, or health (Ritzer 2016).

Economically, globalization has facilitated market growth and capital spread, which has often reduced inequality within nations; for example, the emergence of China to a global manufacturing hub has lifted millions of people out of poverty. However, it also has the potential to paradoxically widen the gap between nations. For example, in parts of Africa where some local industries struggle to compete with cheaper imported goods, leading to job losses.¹ Culturally, globalization has led to both cultural homogenization and cultural exchange and appreciation. The global expansion of American fast food chains, exemplified by McDonald's, has led to cultural homogenization in countries around the world, overshadowing local culinary traditions with global brands (Watson 2006). In contrast, the worldwide dissemination of yoga, which originated in India, promotes an appreciation for Indian culture through physical exercise and spiritual practice (Singleton 2010). Politically, globalization is uniting regions through initiatives like, for example, the European Union's *General Data Protection Regulation*, but it is also widening geopolitical divides, particularly in technology and data governance conflicts, e.g., between the U.S. and China. The impact on the environment is similarly complex, as globalization contributes to both the spread and mitigation of environmental problems. An example of this can be seen in the spread of invasive species, as well as in the global collaboration seen in the Paris Agreement to combat climate change. In social and health terms, the rapid spread of movements such as *#MeToo* and diseases such as *COVID-19* illustrate the role of globalization in spreading norms and challenges that require global cooperation for effective responses.

¹ <https://unctad.org/publication/trade-and-development-report-2019> last visited on: 04.03.2024.

Views on globalization vary widely, from those who see it as a source of all problems to those who view it as a potential solution to many of the world's challenges (Jain 2023). In this era of change, *digitalization* is proving to be both a product and a facilitator of globalization, solving some problems while creating new ones (Weymouth 2023). Digitalization refers to the use of digital technologies to change a business model and create new opportunities for revenue and value creation. It involves converting to a digital company. The objectives of commercial enterprises include increasing market share, sustainability, efficiency, quality, and profitability.

Manually managing the amount of information and its dependencies in companies is often no longer feasible. This necessitates high-quality and high-quantity business processes. Furthermore, there are documentation obligations and regulatory requirements both within and outside the company.

Technical solutions in the field of IT can support, replace, or revolutionize many of the business processes aimed at achieving these economic goals. The utilization of IT for innovative solutions is necessary, particularly in the areas of automation (Industry 4.0), data analysis, robotics, virtual reality (VR) augmented reality (AR) and mixed reality (MR), i.e., extended reality (XR), as well as correlation and pattern recognition with or without artificial intelligence.

Unlike in the past, IT is no longer solely used for supporting and optimizing businesses, but has become an integral part of them. To facilitate the implementation and optimization of such IT systems, business informatics has emerged as an interdisciplinary subject that combines business administration and computer science. Mertens et al. (2023) describe how this field offers more than just the intersection of corporate strategy and information processing. They highlight the use of special methods for coordinating these two disciplines.

A distinction is made between behavior-oriented and design-oriented business informatics. The former aims to discover cause-and-effect relationships, while the latter deals with instructions for the construction and operation of information systems, innovations in the information systems themselves, and the construction of information systems. Design-oriented business informatics serves as a bridge between the economic objectives of business administration and the technical possibilities of computer science, thereby supporting the realization of economic goals, such as increasing market share and profit, or advancing sustainability, efficiency, and quality—see Fig. 1.1. It includes activities such as enterprise modeling, process simulation and optimization, information system design and development, and IT management and governance (Oesterle et al. 2011).

Various frameworks and methods exist for analyzing problems and developing solutions. The goal is to create artifacts using recognized methods. Due to the ongoing emergence of new problems and the fact that many existing problems have not yet been solved by digitalization and/or globalization, it is not possible to address the entire problem space. However, we can focus on the detailed solution and optimization of specific aspects.

As previously mentioned, enterprise modeling, and conceptual modeling in general, are crucial for analyzing, planning, and documenting business aspects such

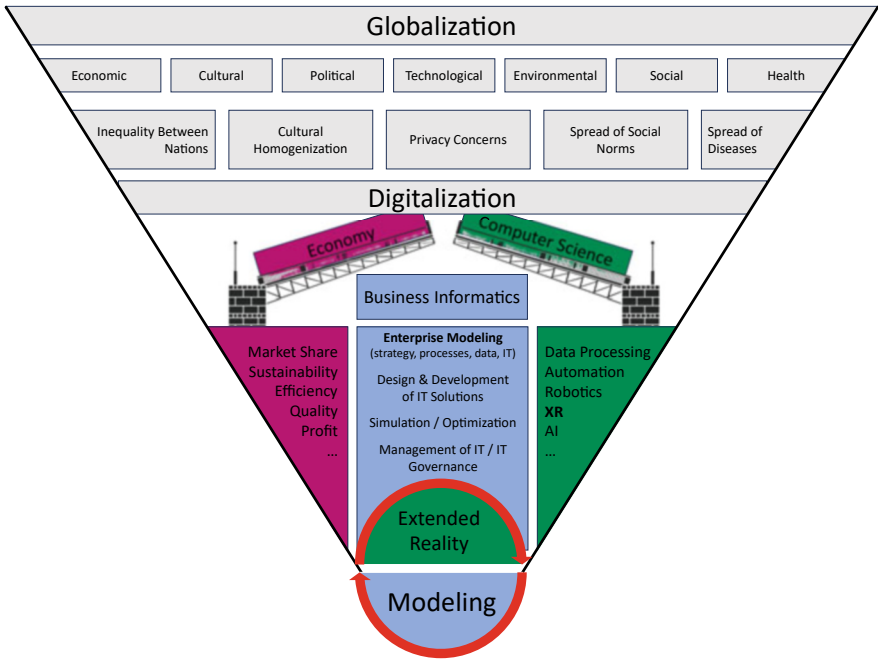


Fig. 1.1 Alignment of this work within the context of globalization, digitalization, and business informatics

as business processes or IT architectures. In addition, emerging technologies such as extended reality, and artificial intelligence offer novel solutions to both, existing and new challenges.

This book, therefore, deals with a specific aspect of business informatics and tries to generate fundamental insights into combining extended reality with conceptual modeling, and particularly the overarching discipline of metamodeling, thereby examining how virtual and augmented reality technologies can affect this sub-area. This introduction provides a first overview of the background, the aim, and the scope of this book. In addition, the chapter summarizes the research methods used and the author's publications in regard to this book.

The chapter is structured as follows: Sect. 1.1 describes the background and motivation for this work, followed by the research objectives and research questions associated with this dissertation (Sect. 1.2). Section 1.3 explains the details of the research methodology that was used during the course of the research. Finally, Sects. 1.4 and 1.5 describe the structure of the book and the intermediary published works related to it.

1.1 Background and Motivation

Augmented reality, virtual reality, and mixed reality, commonly referred to as extended reality (Doerner et al. 2022, p. 21), are technologies that have gained importance in research and industry in recent years (de Souza Cardoso et al. 2020). However, they are not new concepts. As early as the mid-1960s, Sutherland (1965) attempted to define virtual reality as a window through which a user can perceive an artificial, virtual world as if it were a real environment that looks, feels, and sounds like the real world. At that time, three-dimensional (3D) applications required substantial computing power. Thus, high-end supercomputers were needed. In addition, the development of these applications was complicated due to the low-level software platforms available at the time.

Technological progress in recent years has led to the widespread availability of affordable and mobile XR devices that allowed for the broad application of the technology (Yin et al. 2021), e.g., for gaming, navigation, military use, maintenance tasks, or training (Cipresso et al. 2018; Grambow et al. 2021). Different studies highlight the potential of the use of virtual and augmented reality in industry. According to a Gartner study of 2021, the potential of augmented reality is very high, as it will change the way people interact with the real world (Nguyen 2021). A study from PricewaterhouseCoopers (PwC) estimates that VR and AR will deliver an enormous boost to the global economy until 2030 (Dalton and Gillham 2019). Furthermore, a study from 2022 indicates that a majority of US executives are highly interested in exploring VR and AR as a foundation for the metaverse (PricewaterhouseCoopers 2022).

Such immersive and interactive 3D XR applications allow users to participate in experiences that are either very difficult or impossible in real life, or to enhance the real world in ways that would not be possible in reality. For example, the application could provide access to a microscopic world, a fantasy realm, a distant planet, or an expedition into an erupting volcano. Alternatively, it could display the inner workings of the car engine that the user is repairing. Through such applications, users can enter a virtual environment or augment the real world, which can be manipulated to varying degrees and explored in real-time. For enabling such experiences, VR and AR applications use special devices that address the basic senses, i.e., seeing, hearing, and touching. Due to their potential to enhance and complement the learning process, they are increasingly being used in many areas (Mütterlein 2018).

Conceptual modeling, and more generally metamodeling, are well-established approaches for abstracting knowledge from the real world into various forms of models, such as formal representations or visual drawings. These models inherently capture knowledge from the real world, either from existing things or things that might exist in the future. A vision that has recently emerged regarding conceptual modeling, respectively, enterprise modeling, states that modeling will be integrated into daily work practices in the coming years (Sandkuhl et al. 2018). This means that people engage in modeling without noticing it and it becomes a common

practice, just like the use of office applications today. To achieve this vision, multiple challenges must be addressed in research, including adequate model formats, the context of stakeholders, or the scope of models. This encompasses the *stakeholder viewpoint*, the *presentation* and *representation* of models, the models' *scope*, the models' *concerns*, the models' *processing* and *quality*, and the models' *lifecycle*.

In perspective to *stakeholders*, more research is needed on how to improve the social legitimacy of models, i.e., how to make light-weight model creation acceptable and common in a community rather than just among lead users. According to *model representation*, it has to be investigated how everyday work happens and which situations are adequate for model creation and use. To ensure that the right content is represented in the correct way for each actor, the *scope of models* must be controlled. Further research in the *model concern dimension* has to ensure that the concerns supported by modeling methods are exhaustive and sufficient. In the dimension of *processing* of models there are hardly any possibilities to combine modeling with the daily used information systems. Thus, more research is needed to embed modeling-like functionalities in tools which are originally not related to modeling. Regarding the *quality* of models there must be research to find out which quality criteria are too constrictive to enable modeling for non-experts in everyday work and which are so important that they cannot be discarded. Lastly, in the *lifecycle* dimension there must probably be a change of view, since the lifecycle of a model could change with the participation of multiple stakeholders on the same model.

In addition, conceptual modeling detaches the knowledge about the real world from the real world, making it difficult to transfer this knowledge back to reality when needed. For example, processes for assembling guidance in industry are mostly reduced to textual description and two-dimensional (2D) drawings, thus making it difficult to imagine the next step in the real-world environment.

One potential solution to many of the described problems could be the use of extended reality in combination with conceptual modeling. This could reduce or even eliminate some of the barriers that prevent people from using conceptual modeling in their daily work. For example, work instructions, such as assembly processes or machine maintenance instructions, which are typically two-dimensional and disconnected from the actual process, could be connected back to the real world. Virtual information, such as 3D objects, about the next step in the process could be visualized directly for the user in the right place in the real world, at the right time, thus making traditional 2D paper manuals unnecessary. In addition, entire workforce learning processes could be reduced to a minimum by guiding users step by step through work procedures in virtual or augmented reality based on conceptual knowledge from models. This could reduce the problem of skills shortages and allow non-experts to do work that would normally require expert knowledge.

But it is not only the execution of work that could be facilitated by combining conceptual modeling with virtual and augmented reality. It could also revolutionize the way expert knowledge is elicited. Traditionally, domain experts are not modeling experts. This makes it difficult to elicit their knowledge as conceptual knowledge, e.g., in the form of process models. As a result, workshops with modeling experts

are often required, which can lead to misunderstandings and high costs. By using augmented reality, the elicitation of conceptual knowledge could be automated, and knowledge could be documented without the need for the domain expert to know conceptual modeling.

Whether the visual representation of conceptual knowledge in the form of explicit visual models as we know it today would still be necessary, or whether there would be an entirely different approach, remains a topic for further investigation.

An examination of possible use cases indicates two primary directions for the application of virtual and augmented reality in relation to conceptual modeling. First, the use of functionalities of **VR** and **AR** for modeling itself, and second, the incorporation of information from the model space into **VR** or **AR** applications. This second direction includes both design-time and run-time aspects, i.e., the modeling and model-driven generation of **VR/AR** applications and the fueling of model contents into existing **VR/AR** applications. For some of these aspects, approaches have been proposed in academic research, e.g., Campos-López et al. (2021) or Wild et al. (2020). However, almost all of these approaches are very specific for one use case. Therefore, it would be interesting to synthesize the common concepts of these proposed approaches to provide a general approach to solving the problems addressed.

A well-known method for generalizing concepts in the discipline of conceptual modeling is metamodeling (Karagiannis and Kühn 2002). With metamodeling, one can create computer models in predefined conceptual modeling languages such as BPMN,² UML³ or approaches for design thinking, simulation and many more. By providing a platform that defines the concepts and mechanisms for all underlying metamodels, i.e., models of modeling languages, the synergies and common concepts can be defined once and used in common. Furthermore, by using metamodeling and the underlying concepts, different modeling languages can be interconnected and used together, which would not be possible in specific modeling environments. Thus, data between models can be exchanged, and models can be processed. In addition, by providing a general approach, the adaptation and creation of modeling languages is much faster and more productive than without such an approach. Examples of such platforms are ADOxx (Fill and Karagiannis 2013), or MetaEdit+ (Kelly et al. 1996).

Metamodeling platforms define many concepts to solve various problems in conceptual modeling. All of these platforms consider modeling in traditional **2D** space. To the best of the author's knowledge, no one has addressed the new concepts needed on metamodeling, i.e., on the meta²-level, to enable conceptual modeling combined with extended reality as envisioned above. Thus, this dissertation explores and conceptualizes the combination of metamodeling and extended reality. The research carried out is considered ground research.

² <https://www.omg.org/spec/BPMN/2.0/> last visited on: 01.03.2024.

³ <https://www.omg.org/spec/UML/2.5/About-UML/> last visited on: 01.03.2024.

1.2 Research Objectives and Questions

This book will address the question of how metamodeling can be utilized “in” and “for” augmented and virtual reality, i.e., in extended reality. As outlined in Sect. 1.1, such a combination can be helpful in several areas, such as education, enterprise modeling, process visualization, simulation, and many more. It can aid in comprehension, decision-making, and potentially impact the use of conceptual models in daily work, as envisioned by Sandkuhl et al. (2018). To answer the central question of how metamodeling can be utilized “in” and “for” extended reality, multiple steps are necessary. In the following, the research questions for this work are developed.

To gain an understanding of the conceptual and technological concepts of extended reality, in general and in the context of metamodeling, it is necessary to analyze the domain and related approaches. Therefore, the following research question (RQ1) is formulated:

- **RQ1: “What are the necessary components and concepts in extended reality in general and in the context of metamodeling?”**

Since metamodeling involves common concepts beyond specific modeling languages, the findings from RQ1 must be considered at the meta²-level. The question arises where the derived concepts resulting from RQ1 have an influence on a meta²-model. Therefore, the second research question is formulated as follows:

- **RQ2: “What components of a meta²-model must be considered to allow the integration of metamodeling for 3D environments in extended reality?”**

After deriving the necessary concepts and requirements on the meta²-level based on RQ2, the question arises whether these concepts can be integrated into an existing meta²-model or whether it is necessary to specify a new meta²-model. Therefore, the third research question is formulated as follows:

- **RQ3: “How can an existing meta²-model be adapted or extended to incorporate 3D and XR features to meet emerging requirements, or what characteristics would define a newly developed meta²-model enriched with 3D and XR capabilities?”**

On the basis of an extended or new meta²-model resulting from RQ3 and other necessary concepts relevant for 3D enhanced metamodeling from RQ1 and RQ2, a conceptual proposal for a 3D enhanced metamodeling platform considering extended reality can be developed. The fourth research question is thus formulated as follows:

- **RQ4: “What are the architectural components of a 3D enhanced metamodeling platform that considers extended reality?”**

After the first conceptualization resulting from RQ4, the technical feasibility can be shown in a prototypical implementation. Furthermore, this prototypical

implementation should be evaluated. Thus, the last research question is formulated as follows:

- **RQ5: “How can a 3D enhanced metamodeling platform considering extended reality be technically realized and evaluated?”**

After developing the research questions in this section, the next section introduces the scientific methodologies used to answer these research questions.

1.3 Methodology

The “Memorandum on Design-Oriented Information Systems Research” (Oesterle et al. 2011) provides an overview of the discipline relevant to this book, laying out a design- or construction-oriented research approach. The approach is also discussed in “Enzyklopädie der Wirtschaftsinformatik” (Frank 2019). Business informatics begins with designing an information system to meet specific objectives within certain constraints. This design-oriented approach results in constructs, models, methods, and instances, including prototypes and productive information systems (Oesterle et al. 2011). Technical terminologies, languages, and concepts are integral products of this research.

To achieve this objective, a process has been developed that can be divided into four phases (Oesterle et al. 2011). These include the *Analysis Phase*, *Design Phase*, *Evaluation Phase*, and *Diffusion Phase*.

In the *Analysis Phase*, problem descriptions are presented and research questions are defined. This phase examines the current state of problem-solving approaches in both practice and science, and develops a research plan to improve the necessary artifacts. This can be accomplished using various research methods outlined in a research plan. During the analysis phase of this book, the selected methods will include the *Review* method (Fettke 2006) and *conceptual-deductive* and *argumentative-deductive analysis* (Wilde and Hess 2007). In the *Design Phase*, the goal is to derive artifacts using accepted methods. This book will encompass the creation of various concepts and prototypes (Wilde and Hess 2007). In the *Evaluation Phase*, prototypes are evaluated in part by the methods chosen for the specific approach and by publishing different aspects and concepts of the work in intermediate publications. In the final stage, the *Diffusion Phase*, the objective is to achieve the maximum dissemination of the research findings. This will be accomplished through intermediate publications and the publication of this book. An overview of the research inquiries and scientific approaches concerning the various stages of the “Memorandum on Design-Oriented Information Systems Research” (Oesterle et al. 2011) can be found in Table 1.1.

Regarding Information System research and business informatics, the “Memorandum on Design-Oriented Information Systems Research” aligns well with the design science research (DSR) methodology for information systems research

Table 1.1 Research questions and scientific methodologies assigned to the phases of Oesterle et al. (2011)

Phase	Research question	Method
Phase 1: Analysis	RQ1; RQ2; RQ3	Structured literature review;
		Conceptual-deductive analysis;
		Argumentative-deductive analysis
Phase 2: Design	RQ4; RQ5	Prototyping
Phase 3: Evaluation	RQ5	Intermediate publications;
		Empirical user study
Phase 4: Diffusion	RQ1; RQ2; RQ3; RQ4; RQ5	Intermediate publications;
		Final publication

introduced by Peffers et al. (2008). This approach is often applied in construction-oriented research.

In **DSR**, knowledge and understanding of a design problem is gained through the creation and application of an artifact. Hevner et al. (2004) derived seven guidelines for design science in information systems research, which state that the created artifact must address an important, relevant, and previously unsolved organizational problem. However, the specifications for creating the artifacts are not defined, allowing for versatility.

The results of the **DSR** should offer objective and verifiable contributions to the relevant area. Typically, in **DSR** projects, the contribution is the artifact itself, but the contribution can also be an extension and enhancement of the existing knowledge base or the creative development and use of evaluation methods (Hevner et al. 2004). The artifact’s design involves a search process to find a solution to a defined business problem, utilizing existing knowledge and methods to achieve the desired outcome. Finally, research results and contributions must be effectively presented to appropriate audiences (Hevner et al. 2004; Peffers et al. 2008).

Both methodologies described above aim to address real-world problems through the creation and evaluation of IT artifacts, but differ in their structural composition and emphasis. The *memorandum* is structured into four phases: *Analysis*, *Design*, *Evaluation*, and *Diffusion*. In contrast, the **DSR** methodology is composed of six distinct phases: *Problem identification and motivation*, *definition of the objectives of a solution*, *design and development*, *demonstration*, *evaluation*, and *communication*. A comparison of these methodologies shows that most of the phases of the memorandum correspond to the phases of the **DSR** methodology, providing insights into the complementary nature of these frameworks in guiding information systems and business informatics research—see Table 1.2.

This monograph presents nearly four years of research in a particular area. The research was not deterministic from the beginning, and many small research projects were conducted during this time period, all of which are related to the research question described in Sect. 1.2. This book follows the **DSR** methodology as a guide framework, consisting of multiple smaller parts that sometimes employ the **DSR** methodology or other research methodologies. The overall idea of the

Table 1.2 Relation of the phases of the memorandum on design-oriented information systems research (Oesterle et al. 2011) and the phases of the design science research methodology Peffers et al. (2008)

Phases of memorandum	Phases of the DSR methodology
Analysis	Problem identification and motivation
	(Partial) Define of the objectives for a solution
Design	Design and development
Evaluation	Evaluation
	(Partial) Demonstration
Diffusion	Communication

memorandum on design-oriented information systems research is always adhered to. In the following, the outline of the book will be shown.

1.4 Outline

This book is composed of different chapters. In the following, the different chapters are briefly described:

- **Chapter 1—Introduction:** This chapter introduces the topic by providing background information and outlining the research objectives, questions, methodology and structure.
- **Chapter 2—State-of-the-Art and Related Work:** This chapter delves into the existing literature and developments in the field. It covers various aspects of modeling, such as conceptual, enterprise, and metamodeling, as well as extended reality, virtual reality, augmented reality, and the metaverse, with a discussion of both technical and non-technical viewpoints. In addition, the chapter contains an extensive literature study on pairing conceptual modeling with virtual and augmented reality.
- **Chapter 3—Derivation of Generic Requirements for Metamodeling for Extended Reality:** The third chapter presents the generic requirements for metamodeling for augmented and virtual reality by systematically deriving use cases for joining AR and metamodeling, discussing the morphological schemes for the derivation, and providing examples from different perspectives such as strategic, business, and IT.
- **Chapter 4—Specific Requirements for Metamodeling for Extended Reality:** This part identifies specific requirements for integrating metamodeling with XR, such as coordinate mappings, visualization of model components, detection and tracking, context, or interaction.
- **Chapter 5—ARWFMM: A Modeling Method as an Example for Knowledge-Based Virtual and Augmented Reality:** Chapter 5 introduces a new domain-specific visual modeling language for creating augmented reality

scenarios, particularly within the context of metamodeling. This includes the analysis of related approaches, an introduction of existing AR platforms and metamodeling platforms, as well as the specification and evaluation of the new modeling method.

- **Chapter 6—M2AR: An Architecture for a 3D Enhanced Metamodeling Platform for Extended Reality:** This chapter outlines the conceptualization and design of a 3D enhanced metamodeling platform considering extended reality, detailing its structure, components, and the interconnection of its elements. The chapter proposes a conceptual architecture for the platform, integrating various modules for a cohesive 3D enhanced metamodeling environment. Finally, it addresses the practical implications and considerations for implementing the new proposal.
- **Chapter 7—Prototypical Realization of the M2AR Metamodeling Platform:** This chapter presents the initial implementation of the various components of the modeling platform that were conceptualized in Chap. 6.
- **Chapter 8—Evaluation of the M2AR Platform Prototype:** This chapter evaluates three different aspects of the newly introduced metamodeling platform. First, it includes a comparative evaluation of the global generic- and specific requirements, against the first implementation of the metamodeling platform *M2AR*, and against the first implementations of the *ARWFMM* language implemented on *M2AR*. This is followed by a demonstration of *M2AR* and its *ARWFMM* implementation, and third, an empirical evaluation of the comprehensibility of the *ARWFMM* and its language concepts.
- **Chapter 9—Summary and Outlook:** This chapter concludes this book by an alignment with the initial research questions, discussing limitations, providing an outlook for further research, and a final summary.

1.5 Research Contributions

This section presents a list of publications authored or co-authored by the author of this book during and after his Ph.D. research, including authors, title, and abstract. The publications were either presented at international conferences or workshops, or submitted as journal papers.

- **Muff, Fabian; Fill, Hans-Georg (2024): M2AR: A Web-based Modeling Environment for the Augmented Reality Workflow Modeling Language (Muff and Fill 2024b):** This paper introduces M2AR, a new web-based, two- and three-dimensional modeling environment that enables the modeling and execution of augmented reality applications without requiring programming knowledge. The platform is based on a 3D JavaScript library and the mixed reality immersive web standard WebXR. For a first demonstration of its feasibility, the previously introduced Augmented Reality Workflow Modeling Language (ARWFML) has been successfully implemented using this

environment. The usefulness of the new modeling environment is demonstrated by showing use cases of the ARWFML on M2AR.

- **Muff, Fabian; Fill, Hans-Georg (2024): Multi-Faceted Evaluation of Modeling Languages for Augmented Reality Applications—The Case of ARWFML (Muff and Fill 2024c):** The evaluation of modeling languages for augmented reality applications poses particular challenges due to the three-dimensional environment they target. The previously introduced Augmented Reality Workflow Modeling Language (ARWFML) enables the model-based creation of augmented reality scenarios without programming knowledge. Building upon the first design cycle of the language’s specification, this paper presents two further design iterations for refining the language based on multi-faceted evaluations. These include a comparative evaluation of implementation options and workflow capabilities, the introduction of a 3D notation, and the development of a new 3D modeling environment. On this basis, a comprehensibility study of the language was conducted. Thereby, we show how modeling languages for augmented reality can be evolved towards a maturity level suitable for empirical evaluations.
- **Muff, Fabian; Fill, Hans-Georg (2023): A Domain-Specific Visual Modeling Language for Augmented Reality Applications Using WebXR (Muff and Fill 2023c):** Augmented reality (AR) is a technology that overlays digital information onto real-world objects using devices like smartphones, tablets, or head-mounted displays to enrich human comprehension and interaction with the physical environment. The creation of AR software applications requires today advanced coding skills, particularly when aiming to realize complex, multifaceted scenarios. As an alternative, we propose a domain-specific visual modeling language for designing AR scenarios, enabling users to define augmentations and AR workflows graphically. The language has been implemented on the ADOxx metamodeling platform, together with a software engine for running the AR applications using the W3C WebXR Device API for web-based augmented reality. The language and the AR application are demonstrated through a furniture assembly use case. In an initial evaluation, we show, via a comprehensive feature comparison, that the proposed language exhibits a more extensive coverage of AR concepts compared to preceding model-based approaches.
- **Muff, Fabian; Fill, Hans-Georg (2023): Past Achievements and Future Opportunities in Combining Conceptual Modeling with VR/AR: A Systematic Derivation (Muff and Fill 2023d):** Despite the increased interest in virtual and augmented reality in recent years, they are not yet mainstream technologies for everyday use in industry. We argue that a promising approach to facilitate the application of virtual and augmented reality is to combine it with conceptual modeling. In this paper, we thus conducted a systematic literature review on the combination of conceptual modeling with virtual and augmented reality within the last two decades. For this purpose, we reverted to a manual literature search, computational topic modeling, and an expert-driven classification process. This analysis highlights the areas in which such a combination of virtual and

augmented reality and conceptual modeling already exists, as well as the aspects that are not yet covered or that would offer opportunities for further research.

- **Muff, Fabian; Fill, Hans-Georg (2022): Use Cases for Augmented Reality Applications in Enterprise Modeling: A Morphological Analysis (Muff and Fill 2022b):** With the more-widespread availability and cost effectiveness of advanced computer vision technologies, first attempts have recently been made for applying augmented reality in enterprise modeling. Despite these first steps, a systematic analysis of the potential opportunities of this technology for enterprise modeling has so far not been conducted. Therefore, we describe in this paper the results of a morphological analysis that has been performed in a series of expert workshops for deriving according use cases. Based on the technological dimensions of augmented reality and the traditional dimensions of enterprise modeling, we show the potential of this combination by means of three selected use cases.
- **Muff, Fabian; Fill, Hans-Georg (2022): A Framework for Context-dependent Augmented Reality Applications Using Machine Learning and Ontological Reasoning (Muff and Fill 2022a):** The concept of augmented reality permits to embed virtual objects and information within the real context of a user. This is achieved using various sensors to assess the current state of the environment and thus derive the artificially generated information for the user through visual means. For determining the current situation of a user based on sensor data and deriving according actions for information display, we describe a framework that combines machine learning services for object recognition with ontological reasoning. For demonstrating its feasibility, the framework has been prototypically implemented using the Microsoft HoloLens2 AR device and applied to a use case in the domain of work safety measures. Thereby we revert to business process models that have been annotated with concepts from an ontology for letting users specify the situations and actions in work safety scenarios, which can subsequently be processed using objects identified in the real environment of the user and classified based on the concepts in the ontology.
- **Muff, Fabian; Fill, Hans-Georg; Kahlig Eleonora; Kahlig, Wolfgang (2022): Towards Context Dependent Legal Visualizations (Muff et al. 2022a):** In order to understand and assess legal situations in daily life, in-depth knowledge of the law or the availability of legal experts is required. The field of legal visualization has a long tradition in the graphic representation of legal situations. This allows for explanations of legal norms and concrete facts in a form that can be understood by laypersons. This paper explores the use of Augmented Reality (AR) technology in legal visualization. With the help of this technology, users can be presented with legal visualizations for an automatically determined context, helping them to better understand the legal situation. To assess the technical feasibility, a prototype AR application has been developed that can superimpose context-dependent model-based legal visualizations on the real environment. A case study from the field of tenancy law is used to describe the application in practice.

- **Muff, Fabian; Fill, Hans-Georg (2021): Initial Concepts for Virtual and Augmented Reality-based Enterprise Modeling (Muff and Fill 2021a):** One current challenge in enterprise modeling is to establish it as a common practice in everyday work instead of its traditional role as an expert discipline. In this paper we present first steps in this direction through virtual and augmented reality-based conceptual modeling. For this purpose we developed a novel meta-metamodeling framework for virtual and augmented reality-based conceptual modeling and implemented it in a prototypical tool. This permits us to derive further requirements for the representation and processing of enterprise models in such environments.
- **Muff, Fabian; Fill, Hans-Georg (2021): Towards Embedding Legal Visualizations in Work Practices by Using Augmented Reality (Muff and Fill 2021b):** In this paper we outline how legal visualizations can be embedded into every day work practices by using the technology of augmented reality. In brief, augmented reality permits to merge virtual visual representations with the real-world and thereby augment visual perception by additional information and new forms of interaction. For a first conception, we regard the aspects of context, content and interaction to describe which aspects need to be considered for legal visualizations if they are transitioned to augmented reality environments. For illustrating these aspects, we describe a first sample application. The paper concludes with an outlook on the next steps for research on this topic.

In addition, a list of articles authored or co-authored by the author of this book, which were not directly related to the project, but have played a crucial role in shaping the author's perspective, is included.

- **Muff Fabian; Fill, Hans-Georg (2024): Limitations of ChatGPT in Conceptual Modeling: Insights from Experiments in Metamodeling (Muff and Fill 2024a):** Recent years have seen significant progress in machine learning technology, leading to the development of large language models (LLMs) like ChatGPT and Bard, which are currently being investigated in various fields. LLMs already play a role in conceptual modeling research. In this context, we describe insights we gained from experiments for analyzing metamodels using large-language models. The goal of the experiments was to assess to what extent large language models such as used by ChatGPT-4 are able to aid in the processing of state-of-the-art metamodels. In this context, we were particularly interested in whether an LLM could sufficiently understand a complex language definition as used in conceptual modeling tools, and what limitations it would face.
- **Fill, Hans-Georg; Muff Fabian (2024): Bridging the Mental and the Physical World: Conceptual Modeling and Augmented Reality (Fill and Muff 2024):** Whereas conceptual modeling is today widely used for representing knowledge for the purpose of communication and understanding, the combination with augmented reality technologies permits for the first time to anchor this knowledge formally to objects in the physical world using electronic means. In addition, conceptual modeling may help to support the design of complex augmented

reality applications and thus enable non-technical users to better engage with this technology. In this chapter, we thus explore the combination of conceptual modeling and augmented reality by focusing on the role of the subject and how its perception is augmented using augmented reality technologies. From this, we derive two directions in the form of a. Augmented Reality-based Metamodeling and Modeling, and b. Knowledge-based Augmented Reality and illustrate them with recent examples.

- **Crevoiserat, Sophie; Muff Fabian, Fill; Hans-Georg (2023): Towards Augmented Reality Applications for IT Maintenance Tasks based on ArchiMate Models (Crevoiserat et al. 2023):** Augmented reality permits to embed virtual objects in the real environment to enhance the perception of users. In this paper, we describe an approach for embedding conceptual models using augmented reality in IT maintenance scenarios. It is based on the ArchiMate modeling language that has been extended with the goal of bridging the gap to models for physical environments. This allows, for example, to guide users in IT maintenance tasks by displaying necessary information originating from the models in the real world. The approach has been implemented on a novel metamodeling platform, which natively supports augmented reality scenarios.
- **Fill, Hans-Georg; Muff, Fabian (2023): Visualization in the Era of Artificial Intelligence: Experiments for Creating Structural Visualizations by Prompting Large Language Models (Fill and Muff 2023):** Large Language Models (LLMs) have revolutionized natural language processing by generating human-like text and images from textual input and can become a powerful tool for many industries and applications, generating complex visualizations with minimal training. However, their potential to generate complex 2D/3D visualizations has been largely unexplored. We report initial experiments showing that LLMs can generate 2D/3D visualizations that may be used for legal visualization. Further research is needed for complex 3D visualizations and 3D scenes.
- **Muff, Fabian; Spicher, Nathalie; Fill, Hans-Georg (2023): Integrating Physical, Digital, and Virtual Modeling Environments in a Collaborative Design Thinking Tool (Muff et al. 2023):** Design thinking is a creative process that requires brainstorming techniques that take place in a physical environment. However, such physical interactions are not possible in remote environments. In this paper, we propose a software tool for design thinking that bridges the gap between physical, digital, and virtual modeling environments. We describe and evaluate a virtual storyboarding application that enables remote collaborative design thinking in 3D and the conversion of these 3D models into 3D digital models. To evaluate the approach, we conducted an experiment with students and were able to derive directions for further research in this area.
- **Muff, Fabian; Härer, Felix; Fill, Hans-Georg (2022): Trends in Academic and Industrial Research on Business Process Management—A Computational Literature Analysis (Muff et al. 2022b):** An important aspect of enterprise information systems is the management and execution of business processes. For exploring the evolution of topics in business process management in academia and industry, we present the findings from a computational literature

analysis. For this purpose, we revert to the full texts and metadata of the proceedings of the International Conference on Business Process Management and its workshops as a sample. In addition, the data has been enriched with data on the academic or industrial provenance of the authors. For identifying the most important topics in business process management, we performed a content-based analysis of over 1200 papers using Latent Dirichlet Allocation. This analysis gives insights into the development of topics over time and identifies recently emerging topics.

- **Fill, Hans-Georg; Härer, Felix; Muff, Fabian; Curty, Simon (2021): Towards Augmented Enterprise Models as Low-Code Interfaces to Digital Systems (Fill et al. 2021):** Traditionally, enterprise models have been used for representing knowledge on all aspects of an organization. This aided not only in composing a holistic picture of the different layers of an enterprise in terms of its business model, products and services, business processes and IT architecture, but also for describing the interdependencies between the layers. Depending on the degree of formalization, algorithms may be applied to the models, e.g. for simulations. With the upcoming of low-code approaches in software engineering, we regard in this position paper how similar concepts may be integrated in enterprise engineering. In particular we regard augmented enterprise models as interfaces to digital systems and illustrate this view with approaches for semantic technologies, data analytics and blockchain platforms. It is envisaged that such approaches will aid domain experts in integrating digital technologies in their daily work practices.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



Chapter 2

State-of-the-Art and Related Work



This chapter introduces the state-of-the-art in research and industry in the areas relevant to this book. In addition, related work is introduced. The chapter is structured as follows. Section 2.1 introduces the foundations of the modeling area, including conceptual modeling, enterprise modeling, and metamodeling. Section 2.2 presents the foundations for extended reality. This includes introductions to virtual reality, augmented reality, and the metaverse, as well as some distinctions of the different technologies in the context of this work. Section 2.3 discusses related work by showing the finding of a comprehensive literature analysis.

2.1 Modeling

In a generally accepted definition of Stachowiak (1973, pp. 131–133), the term *model* has multiple meanings. On the one hand, the term *model* may be interpreted as an image or an example of something. Additionally, it can also be interpreted as a representation of a specific original. This “something” or “original” can be a real thing or an intended system the model will represent (Kühne 2006). In the context of this work, a model is understood in the following as the replication of a section of reality (an archetype)—its image, or as an image of an intended future system. For the sake of clarity, we denote the term “original” as “subject”. Furthermore, Stachowiak (1973) defines three main features of the general model concept. Representation, abstraction, and pragmatics:

Representation Feature Models are always models of something, namely images, representations of natural or artificial subjects, which themselves can be models. Such subjects can be created in a natural way, produced technically, or given in any other way. They can belong to the realm of symbols, to the world of ideas and concepts, or to physical reality.

Abstraction Feature Models typically do not encompass all features of the subject they represent, but rather only those that appear pertinent to the creators and/or users of the model in question. Gaining an understanding of which attributes of the subject are captured by the model, as well as recognizing that not all subject attributes are captured by the associated model, necessitates comprehension of all attributes of both the subject and the model.

Pragmatic Feature Models are not clearly assigned to their subjects per se. They fulfill their substitution function for certain subjects within certain time intervals and under restriction to certain mental or factual operations. Thus, models are not just models of something. They are also models for someone, which can be a human individual or an artificial model interpreter like a computer. Furthermore, models perform their function in time, e.g., only in a certain interval. Finally, a model has a purpose. In other words, a pragmatically complete definition of the concept of model must take into account not only the question of what subject a model is of, but also for whom, when, and how.

According to Stachowiak (1973, pp. 138–139), models have multiple purposes in science and practice. They serve as demonstration models, illustrating unclear interrelationships. They function as experimental models for the determination or verification of hypotheses. They convey behavior knowledge in a concise form, as theoretical models. Ultimately, they offer decision-making and planning aids. Models are constructed from subjects if these subjects require enlargement or reduction for better understanding. This is especially so when the subject is too distant or not readily accessible, too dangerous to approach, or is too costly to access. Models are also created to help clarify, simplify, or represent complex events. In addition, models help trace multifaceted conditions back to essential basic relationships, enabling their explanation or prediction.

The process of abstraction involves isolating specific characteristics using the mind. During this process, significant properties are emphasized, while unimportant properties are disregarded. The determination of the importance of a property depends on pragmatic considerations and varies according to awareness and interest (Prechtl and Burkard 1999).

In addition to the representation and the abstraction feature introduced above, Kühne (2006) divides the representation into *projection* and *translation*. Thus, projection is a structure-preserving operation that creates a relationship between a model and the original subject. The information that remains after model projection is dependent on the ultimate purpose of the model and its pragmatic usability, i.e., the intended audience and purpose of the model—see pragmatic feature above. Translation, on the other hand, changes the syntactical representation of the original subject to another representation, i.e., the formalization by a modeling language (Kern 2016).

Besides the representation concepts projection and translation, there exist also important forms of abstraction, such as generalization and classification. Classification is a process of grouping elements that are similar in terms of certain characteristics into one category. It involves assigning elements to a particular

type based on their shared properties (Kühne 2006). This involves grouping a large number of individual items into a new entity, usually referred to as a class, which shares the same set of properties, but may have different values for those properties (Kern 2016). The generalization function takes elements that are equivalent according to some relation and maps them to the same model element. This should not be confused with classification, which is aimed at finding a universal for equivalent elements, while generalization is intended to broaden the scope of existing universals (Kühne 2006).

Having introduced the foundations of modeling, we can further dive into the area of conceptual modeling in the next section.

2.1.1 *Conceptual Modeling*

Models, as described at the beginning of the section, and the activity of creating such models (modeling), is a fundamental part of computer science, providing a basis for understanding between developers and users and allowing them to focus on the task at hand without worrying about implementation (Roussopoulos and Karagiannis 2009). Conceptual modeling involves the creation of models that are independent of the technology and strategy used to address a problem. These models are designed to describe the problem without being influenced by the methods used to solve it (Kaschek 2008). The most widely used definition of conceptual modeling is the one by Mylopoulos stating that conceptual modeling is: “the activity of formally describing some aspects of the physical and social world around us for purposes of understanding and communication”, which requires the adoption of a formal notation (Mylopoulos 1992, p. 52). Conceptual models capture relevant aspects of a subject, e.g., a manufacturing workplace and the activities that take place at that workplace. It can serve as a common point of understanding about a subject by means of graphic and linguistic concepts, taking into account the pragmatics introduced in Sect. 2.1. In addition, conceptual models can be valuable to introduce a novice to a given subject.

Since the notation of a conceptual model is formally defined, it is possible to capture the semantic meaning of the model. In general, conceptual models are meant to be used by humans in the first place, not by machines (Mylopoulos 1992). Thus, conceptual modeling must be distinguished from “knowledge representation” and “semantic data modeling”. Even though knowledge representation and semantic data modeling also involve capturing of knowledge about a given subject, they are not synonyms. According to Sowa (2000), the study of knowledge representation involves the utilization of theories and methods from logic, ontologies, and computing to enable machines to carry out “intelligent” activities. Furthermore, it is assumed that the created knowledge bases will be used by another system (Borgida 1990). Semantic data modeling introduces ideas about how conceptual schemata will be implemented on a physical machine. This type of modeling is more

restrictive than conceptual modeling, resulting in a simpler notation that is more suitable for implementation (Mylopoulos 1992).

Conceptual models are essential in many areas of computer science. In recent years, they have been used in different fields, such as designing information systems, representing knowledge for artificial intelligence, modeling organizational structures, business processes, software development processes, software requirements, and simply modeling aspects of the world to facilitate communication and comprehension (Roussopoulos and Karagiannis 2009).

Conceptual modeling can be viewed from different perspectives. One perspective is the modeling process itself, where conceptual models are created to abstract something from the real world, i.e., during *design-time*. Another advantage of creating such models using a well-defined modeling language (cf. Sect. 2.1.3) is that they can be interpreted by computer algorithms during *run-time*, such as for simulating process time. These two aspects are referred to as *interaction aspects*, since they involve interaction with conceptual models.

Furthermore, there are two aspects to consider for *model-based applications*: *design-time* applications for modeling itself and *run-time* applications that take models as input. These aspects are referred to as flexibility aspects, as they improve flexibility and reusability (see Sect. 2.1.3). Figure 2.1 visualizes these different aspects of conceptual modeling. This distinction will become more important in Sect. 2.3.

A specific area of conceptual modeling is the area of enterprise modeling (EM) which will be introduced in the following.

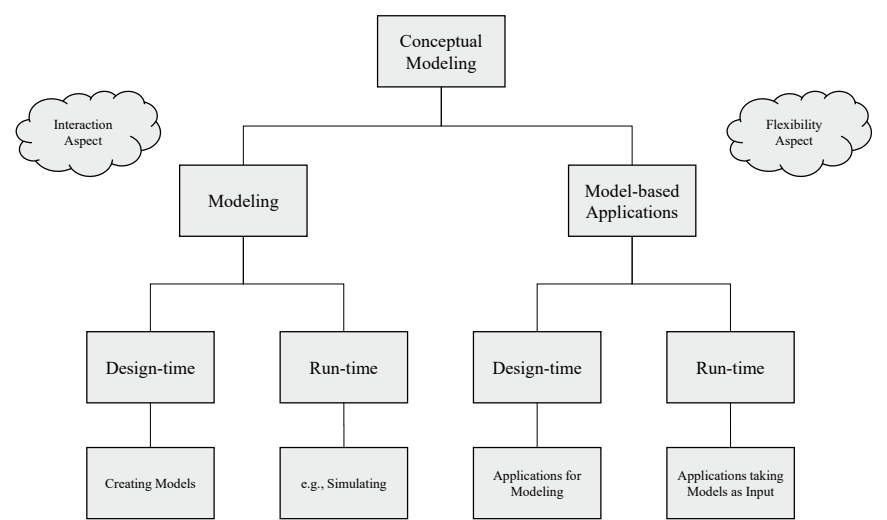


Fig. 2.1 Aspects of conceptual modeling distinguishing between interaction aspects and flexibility aspects

2.1.2 Enterprise Modeling

EM involves the creation of models, which are abstract representations, to help people or machines understand, analyze, (re)design, reason, control, and even learn about different aspects of an enterprise (Sandkuhl et al. 2014; Vernadat 1996, 2020).

No consensus has been reached on a precise definition of EM, however, it is commonly referenced as “a set of activities dealing with representing and describing the structure, behavior, and organization of the whole or part of a business entity” to redesign its structure, processes, and the way it handles its internal and external activities. EM aims to optimize this structure and processes and evaluate their performance to improve the efficiency and effectiveness of the company (Vernadat 1996).

The origins of enterprise modeling can be traced back to the fields of organization sciences, systems theory, and systems engineering. It has been particularly influenced by software engineering, information technology, computer simulation, and industrial engineering. This wide range of requirements has led to the development of a variety of modeling techniques and languages (Vernadat 2020). Thus, EM is closely related to conceptual modeling. Enterprise models are a key component for understanding, analysis, engineering, improvement, optimization, maintenance, and even management and control of enterprise systems. Consequently, EM is fundamental for enterprise engineering, integration and management, helping organizations understand their current state, plan future improvements, and make informed decisions about their operations (Vernadat 2020).

A big area of EM is business process management, including business process modeling. However, EM should not be confused with only that. Other aspects, such as functional, information, resource, organization, or economic aspects, are equally important. They are modeled and analyzed from different angles, e.g., in goal/objective models, functional models, process models, conceptual data models or object class diagrams, resource models, organizational models, factory layout diagrams, structural diagrams, as well as control or sequence diagrams. An enterprise model is therefore the sum of the models obtained in each view (Vernadat 2020).

Since such models are usually created conforming to a formal- or semi-formal modeling language, they can be processed by computers, thus they can be used not only to store information, but also for processing this information, e.g., for simulation.

Research in the field of EM has identified a few key components, such as the modeling procedure or method, the model that results from the modeling activity, the tool support for modeling, and the organizational structures that frame modeling (Sandkuhl et al. 2018).

One possibility to support the discipline of enterprise modeling is the use of metamodeling and language development (Sandkuhl et al. 2018). In the next section, these terms are introduced.

2.1.3 *Metamodeling*

Examination of the term “metamodel” reveals that the prefix “meta” is used when an action is repeated. For example, a conversation about how to have a conversation is a “meta-conversation”, or learning general learning techniques while studying a particular subject is “meta-learning”. In the nineteenth century, mathematicians were concerned with establishing a solid foundation for mathematics, so they used mathematical methods to ensure that ordinary mathematics could be done accurately. This new field was called “metamathematics”, as mathematical methods were applied to mathematics itself. In summary, the prefix “meta” is used before an operation to indicate that it has been applied twice (Kühne 2006). Different definitions of the term “metamodel” exist, e.g., “A metamodel is a model of models.” (Miller and Mukerji 2003), suggesting that a metamodel is a representation of another model (Kühne 2006).

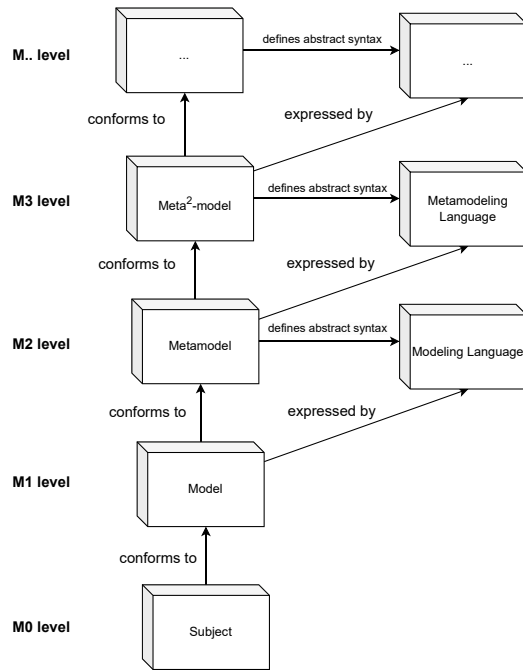
To limit the definition of the term, we will draw upon the perspectives of metamodeling presented in the field of enterprise modeling—see Sect. 2.1.2—and more specifically as discussed in Strahringer (1998) and Karagiannis and Kühn (2002). As will be seen in the following definitions, metamodels in this context are closely related to the domain of conceptual models—see Sect. 2.1.1—and are usually described in a semi-formal manner (Fill 2009).

It is necessary to consider metamodeling from two practical perspectives: The concept of metamodeling and the technical application of it. The objective of the first perspective is to create a high level of abstraction of real-world connections for a particular user, based on semi-formal definitions. This abstraction leads to a simpler comprehension and more efficient control by domain experts. An example of this perspective is the unified modeling language (UML) (OMG 2012). The models created through this high level of abstraction can still be used as a foundation for direct code generation and other deployments, e.g., for model-driven generation of object-oriented class definitions (Fill 2009).

The technical side of metamodeling is explored as well in the literature and by the Object Management Group (OMG), an international standard organization, to create definitions of modeling languages through metamodels and hierarchies of metamodels. The aim of these undertakings is to create a shared collection of items and connections that can be reused in multiple modeling languages (Karagiannis and Kühn 2002; Strahringer 1998). Strahringer (1998) describes these two perspectives as the process of modeling and the concepts of a language.

A metamodel is not only a model of a model, as defined in Miller and Mukerji (2003). A metamodel consists of types or metamodel elements that define a possible set of elements in a model. As described in Sect. 2.1, concept of “classification” increases the abstraction level of a metamodel compared to a model. Metamodeling leads to the creation of a metamodel that outlines the abstract syntax of a language. Thus, in metamodeling, a model “conforms to” a metamodel and is expressed through a modeling language. The definition of a metamodel further requires a language denoted as metamodeling language. As for modeling languages, the

Fig. 2.2 Model hierarchy depicting the relation between model, metamodel, meta²-model, modeling language and metamodeling language. Adapted from Strahringer (1998) and Karagiannis and Kühn (2002). The bottom level contains the subject that should be abstracted by a model (M0). The first level contains models (M1), the level above includes metamodels (M2) and at the top level is a meta²-model (M3)



abstract syntax of a metamodel is defined through abstraction, and thus expressed by a modeling language. Since modeling concepts are applied to a metamodel, this new metamodel is denoted as a meta-metamodel or meta²-model. Analogously to the relation of models and metamodels, a metamodel “conforms to” a meta²-model and is expressed by a metamodeling language. Theoretically, this abstraction process could be repeated infinitely, but after some iterations, further abstraction does not make practical sense. The relationship between model, metamodel, meta²-model, modeling language, and metamodeling language is illustrated in Fig. 2.2.

Subject, Model, metamodel, and meta²-model build a four-level model hierarchy. The bottom level contains the subject that should be abstracted by a model (M0). The first level contains models (M1), the level above includes metamodels (M2) and at the top level is a meta²-model (M3). A model hierarchy possesses exactly one meta²-model that potentially defines many metamodels. Each of these metamodels can further define multiple models that themselves conform to a subject. An example of the model hierarchy of the levels M3–M1 is visible in Fig. 2.3.

The Petri Net model itself is an abstraction of a subject, e.g., a simple process. This model (M1) conforms to a metamodel (M2), i.e., the Petri Net metamodel, and is expressed by the Petri Net modeling language. The Petri Net metamodel comprises the concepts *Transition*, *Place*, and *Arc*. The Petri Net metamodel (M2) conforms to a meta²-model (M3), in this case the ADOxx meta²-model (Fill and Karagiannis 2013), and is expressed by the metamodeling language of ADOxx. The

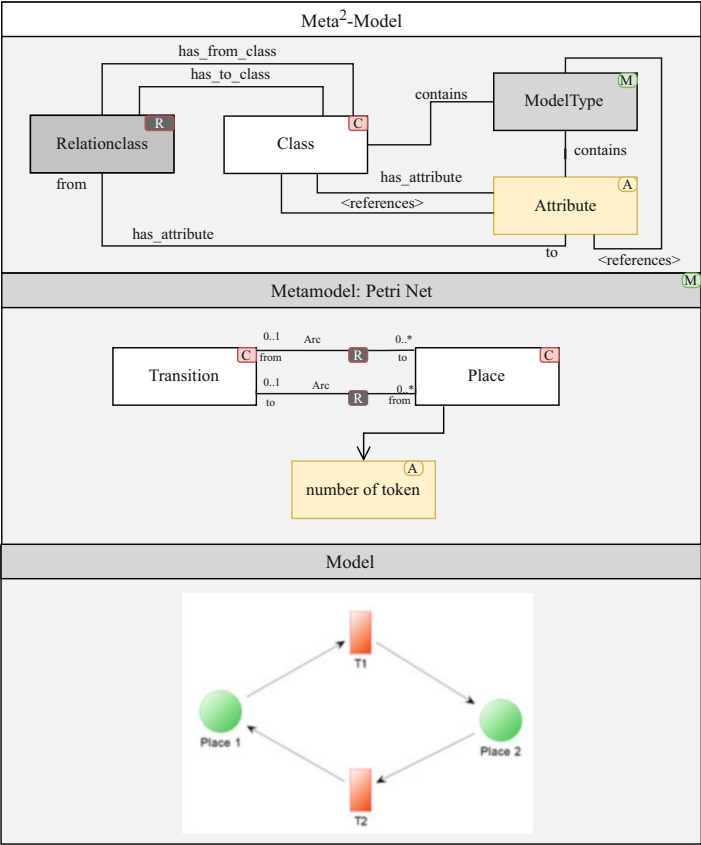


Fig. 2.3 Example of the model hierarchy of the levels. M3: The ADOxx meta²-model model (Fill and Karagiannis 2013). M2: Petri Net metamodel (Petri and Reisig 2008). M1: Example of a Petri Net model

ADOxx meta²-model is depicted here in a very reduced form with the concepts *ModelType*, *Class*, *Relationclass*, and *Attribute*.

There are exceptions of this model hierarchy, where metamodels describe themselves, e.g., the *ECore* metamodel (Steinberg et al. 2009). In the context of this work, we consider the model hierarchy as described above. The discipline of metamodeling contains not only the definition of metamodels according to a meta²-model but also entire modeling methods, which will be the subject of the next section.

2.1.3.1 Modeling Method

According to Karagiannis and Kühn (2002), modeling methods consist of two components: (1) A *modeling technique* and (2) *mechanisms and algorithms*. A modeling technique can be broken down into two components: a modeling language and a modeling procedure; see Fig. 2.4.

2.1.3.1.1 Modeling Language

The modeling language is composed of the elements used to create models and is characterized by its syntax, semantics, and textual or graphical notation. The syntax is described by a grammar that describes the elements and rules for creating models. The semantics of a modeling language is determined by how its syntax is mapped to a semantic schema. The notation describes how to display a modeling language. Static techniques define symbols to represent syntactical components, such as pixel-based graphics, vector graphics, or three-dimensional (3D) models, without taking into account the state of the modeling components while modeling. Dynamic approaches divide the notation into two components. A representation part and a control part. The representation part maps to the static approach, while the control part outlines regulations for querying the model state and altering the representation based on the model state (Karagiannis and Kühn 2002).

2.1.3.1.2 Modeling Procedure

The modeling procedure describes how the modeling language is used to create instances of models, i.e., what steps have to be done to achieve a certain result (Karagiannis and Kühn 2002).

2.1.3.1.3 Mechanisms and Algorithms

Mechanisms and algorithms offer the capability to utilize and assess the models created using the modeling language. Mechanisms can be divided into generic, specific, and hybrid. Generic mechanisms are incorporated in the meta²-model, so they can be employed for all metamodels based on the meta²-model. Specific mechanisms are designed for a particular metamodel. Hybrid mechanisms are implemented on the meta²-model, but are modified to particular metamodels, for example, to improve usability (Karagiannis and Kühn 2002).

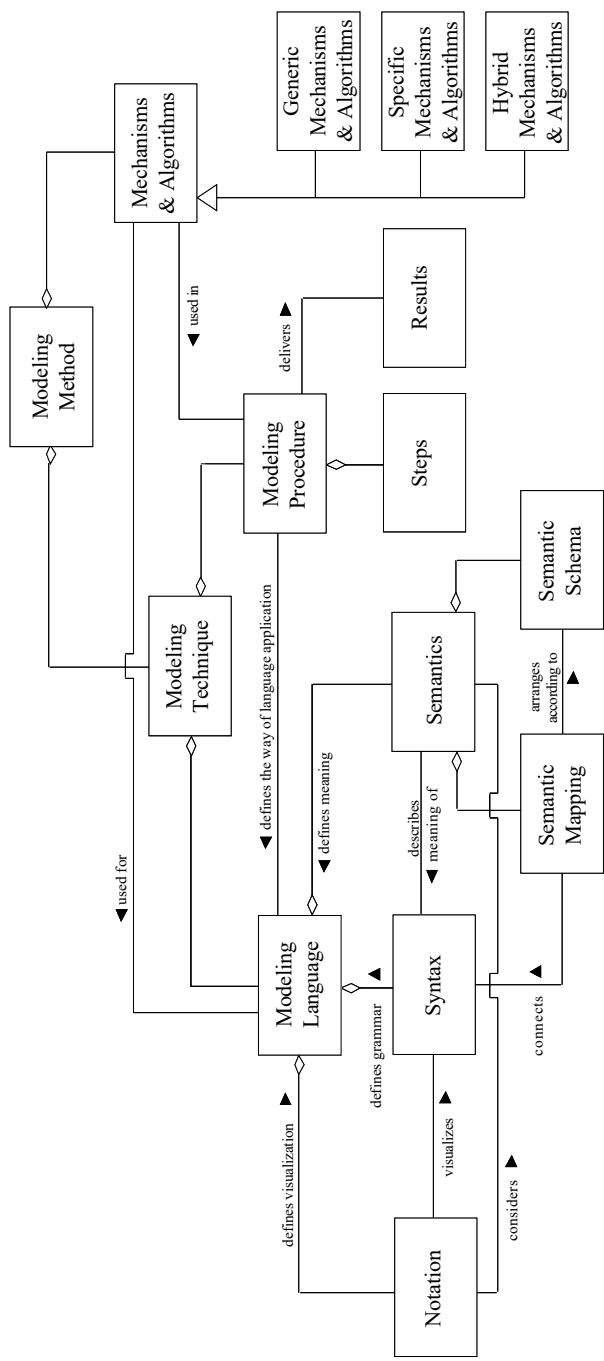


Fig. 2.4 Components of a modeling method from Karagiannis and Kühn (2002). Adapted from Karagiannis and Kühn (2002)

2.2 Extended Reality

In this section, a basic introduction to extended reality technologies is provided. The three terms virtual reality, augmented reality, and mixed reality can be summarized under the generic term extended reality (XR) (Doerner et al. 2022, p. 21). These technologies digitally enhance reality with virtual content to varying degrees, with the aim of integrating digital content into the real or virtual world, enabling interaction with virtual information and, in augmented reality (AR) and mixed reality (MR), the real world.

There are no hard lines for the distinctions between the different categories. It is much more appropriate to think of the various technologies in terms of a continuum between the real environment and the virtual environment. Figure 2.5 shows this continuum according to Milgram et al. (1995) who already tried to classify AR, augmented virtuality (AV), referred to as VR in this work, and MR on a continuum between the fully real environment and the fully virtual environment.

VR is characterized by the complete isolation of the user from reality and a high level of immersion. It is thus at the right end of the continuum. On the other hand, in augmented reality the user sees the real environment at any time. Thus, AR is on the left side of the continuum. Mixed reality is a combination of augmented reality and virtual reality, allowing for both immersive and non-immersive scenarios. To simplify things, we can think of MR as an extended and more interactive version of AR that also involves VR. In the rest of this book, we will talk about virtual and augmented reality, which also includes mixed reality. To clarify the different technologies, the terms VR and AR will be explained in more detail in the following sections.

2.2.1 Virtual Reality

Doerner et al. (2022) summarizes VR as a computer system that consists of the appropriate hardware and software to create the idea of virtual reality. The content represented by the VR system is called *virtual world*.

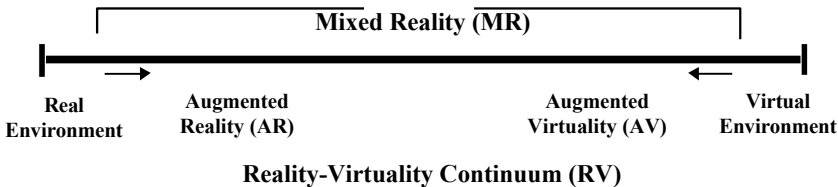


Fig. 2.5 Simplified representation of the Reality-Virtuality Continuum. Adapted from Milgram et al. (1995)

2.2.1.1 Non-technical View on Virtual Reality

The virtual world includes models of objects, their behavioral description for the simulation model, and their arrangement in space. When a virtual world is represented with a **VR** system, we speak of a virtual environment for one or more users. The generation of stimuli is only one task on the way to virtual reality. People in virtual reality not only want to see and feel the world, they also want to interact with the world.

For instance, in a virtual world, a user may encounter a virtual punching bag and have the option to either punch it or avoid it as it rebounds. This requires the simulation of a virtual world with virtual, computer-generated **3D** objects. The human actions must be recognized by the simulation so that they can be incorporated into the simulation. This simulation, in turn, influences the stimuli perceived by the user in the virtual world. If one moves in the real world, this change of position must also be considered in the generation of stimuli in the virtual world. The calculations for this stimulus generation are done by a computer. It is possible not only to simulate a human action, but also to simulate changes in the virtual world that are independent of human actions; e.g., light can be changed by simulating daylight or weather, or simulated wind can move virtual objects.

Although the first approaches to virtual reality head-mounted display (**HMD**) were developed in the 1960s (Sutherland 1968), **VR** has been increasingly researched in recent years, as recent technological advances have made virtual reality affordable through its availability on standard smartphones, tablets and head-mounted displays (Yin et al. 2021).

The purpose of **VR** from a non-technical point of view is to create a virtual immersive world. Most of the time, this virtual world simulates the real world as closely as possible, including simulation of realistic lights and physics. In virtual reality, one is not bound to physical laws. This makes it possible to create fantastic virtual worlds, which are either completely unreal or simply show a past or future world. Scenarios for this would be, for example, the representation of the populated planet Mars or the reproduction of a battle scene in World War II.

2.2.1.2 Technical View on Virtual Reality

There exists no clear and generally accepted definition of virtual reality, but there are some characteristics of **VR** that are widely accepted. The specific type of content input or output is probably a clear feature that all **VR** systems have in common. For example, a helmet with integrated screens on the user's head, special stereo glasses, data gloves, or even an entire room with displays that completely surround the user (Doerner et al. 2022). This creates the possibility of defining **VR** from a technological point of view. The definition does not focus on individual input and output devices, as they can become technically obsolete very quickly. Future proof **VR** definitions should also be compatible with visionary ideas such as Sutherland's *ultimate display* (Sutherland 1965).

The ultimate display would, of course, be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal. With appropriate programming such a display could literally be the Wonderland into which Alice walked. (Sutherland 1965, p. 2)

VR can be characterized as a differentiation from traditional computer graphics. Thus, VR is an extension of the 3D content of computer graphics, particularly in real-time computer graphics. In combination with these 3D graphics, three-dimensional displays are used. For example, with classic HMDs, this is achieved by stereoscopic displays. Often it is not only about the visual sense, but the presentation of the content is multisensory, by also addressing the sense of hearing or touch. In many cases, 3D interaction devices that can be tracked in 3D space are used. All these aspects have the goal of surrounding the user with the virtual world, as summarized in a statement by Steve Bryson in 1993.

Virtual reality (VR) refers to the use of three-dimensional displays and interaction devices to explore real-time computer-generated environments. (Steve Bryson, Call for Participation 1993 IEEE Symposium on Research Frontiers in virtual reality)

Based on this enclosure of the user in the virtual world, immersion is often referred to in the literature as a central feature to distinguish VR from other human-computer interfaces. The goal is to allow the user's sensory impressions to be addressed as comprehensively as possible by one or more output devices. According to Slater and Wilbur (1997) immersion is based on four technical characteristics of the output devices:

(1) Human sensations should be generated by the computer as exclusively as possible, i.e., the user should be isolated from the real environment as much as possible. (2) As many senses as possible should be addressed. (3) The output devices should surround the user rather than provide a narrow field of view. (4) In addition, the output devices should provide a vibrant display, e.g., with high resolution and quality color.

Thus, immersion is not simply present or absent, but may be present in different degrees. For example, a conventional HMD can have a larger or smaller screen, with better or worse resolution. Therefore, it is more or less immersive. Complete immersion is currently a vision that cannot be realized with current technology, but a high degree of immersion can already be achieved through HMDs or rooms equipped with displays (Doerner et al. 2022).

2.2.2 Augmented Reality

Some parts of this section have been published in a similar form as a research paper in the journal *Jusletter-IT* with the title: *Towards Embedding Legal Visualizations in Work Practices by Using Augmented Reality* (Muff and Fill 2021b).

Augmented reality is a technology that allows virtual images generated by a computer to be overlaid with physical objects in real time (Zhou et al. 2008). Unlike virtual reality, the user is not immersed in an artificial world but sees the real world around him. A generally accepted definition of AR comes from Azuma (1997). He describes AR as a technology that combines the real world and virtual imagery, is interactive in real time, and registers virtual imagery with the real world.

2.2.2.1 General Augmented Reality Concepts

Augmented reality is based on three core concepts from the field of computer vision (Schmalstieg and Höllerer 2016): (1) *Detectables/Trackables*, (2) *Coordinate Mappings*, and (3) *Augmentations*. First, to determine the location and orientation of the real-world environment, computer vision algorithms are used to estimate the position and orientation based on two-dimensional (2D) or 3D sensor information, e.g., from a camera stream or a LiDAR scanner (Doerner et al. 2022; Saxena and Verma 2022). This detection can revert to *detectables* in the form of *natural features* or *markers* such as QR codes as surrogates to simplify detection and tracking (Schmalstieg and Höllerer 2016). Coordinate mappings are then needed to align objects in the real and virtual worlds to each other. Thus, a *real world origin reference* position, e.g., stemming from global positioning system (GPS) coordinates, must be mapped to the *global coordinate system* of the virtual environment. Furthermore, the *local coordinate systems* are used for any real-world or virtual object. These allow us to define *reference points* for placing virtual objects relative to other objects, independent of the current global coordinates. Finally, virtual information is superimposed on the real world through so-called *augmentations*. These can be animations, 2D images, videos, audio, text labels, 3D objects, hyperlinks, checklists, or forms. By defining *anchors*, which are a sort of *reference points*, augmentations can be fixed at a particular position in real space.

For more complex AR scenarios, further concepts are necessary. This includes, in particular, the integration and processing of additional data that are acquired throughout the life-cycle of an AR scenario via sensors or user interactions. To enable dynamic changes in the AR environment, at least basic workflow concepts such as *triggers*, *conditions*, and *actions* need to be foreseen (Wild et al. 2014). Thereby, triggers include: click, detection, sensor, or timer events, voice commands, entry/exit of defined spatial areas, or gestures. Conditions specify the branching of different process flows, while actions refer to any changes applied to virtual objects, such as their appearance, disappearance, or transformations, e.g., rotation, scaling, and positioning.

2.2.2.2 Applications

The application of augmented reality has been explored in various areas such as personal information systems, industrial and military applications, medical

applications, [AR](#) for entertainment or [AR](#) for the office (van Krevelen and Poelman 2010). If we look at [AR](#) environments on a functional level, they can be divided into two broad categories (Hugues et al. 2011): The augmented perception and the artificial environment. The first category emphasizes that [AR](#) provides a decision support tool. It can provide information that enables a better understanding of reality and ultimately optimizes our actions in relation to reality. This functionality can be divided into further sub-functionalities, ranging from information enhancement to the replacement of reality by virtual objects. This allows visualization of objects and relationships that are only recognizable through [AR](#). The second category is artificial environments. This means that environments do not represent reality as it is perceived, but augment it with information that is not perceptible but could exist in a future or past reality. An example would be the [3D](#) visualization of a historic building that collapsed a long time ago.

2.2.2.3 Technical Components

If we look at the components of [AR](#) at a technological level, we can divide the technology into different electronic sensors for the input and output of information and components for processing this information. The output information can be visual, acoustic, or haptic, with visual output being certainly the main part of [AR](#). For information input, different sensors are needed to make the overlap of real and virtual objects in [AR](#) as natural as possible and to sense and monitor the environment. These sensors must be able to detect motion, orientation, and objects in the environment. For motion sensors, we can distinguish between acceleration, i.e., linear motion on the x , y , or z axes, and rotational motion on all [3D](#)-axes. Orientation can be calculated from a fixed magnetic point, i.e., typically magnetic north. In combination, such sensors help to track the motion and orientation of the [AR](#) device, and thus the user, and adjust the representation of the virtual objects. The environment is composed of objects in real space, their depth mapping, as well as the light conditions or color mapping. This is typically detected through one or more camera sensors, whose information is processed accordingly. If all these components are present, a realistic combination of the real world and virtual objects can be achieved (Muff and Fill 2021b).

2.2.2.4 Output Devices

To enable [AR](#) functionality, there are several types of output devices. Different sensors (Ω) can be used to capture the reality (α) and the user's environment. Then an output device is used to display an image of the real environment (α') and the additional virtual objects (β) on a display (μ). In addition to virtual objects, information (π) can be projected onto real or virtual objects (see Fig. 2.6).

More specifically, there exist devices with a screen, for example, a smartphone or a [MR](#) headset. These have integrated cameras and other sensors to capture the

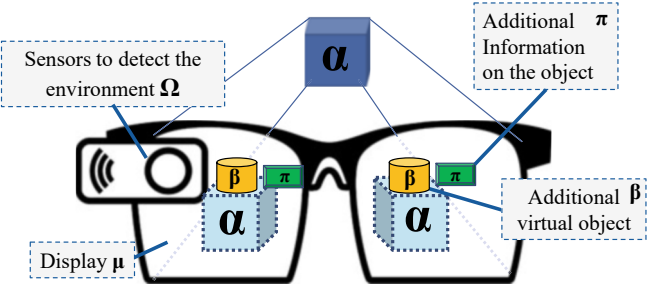


Fig. 2.6 Conceptual components of an AR environment. According to concepts from Muff and Fill (2021b)

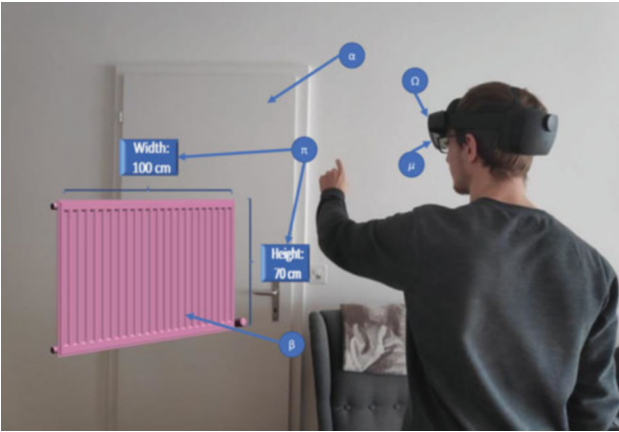


Fig. 2.7 Exemplary illustration of the conceptual components of an AR environment. Adapted from Muff et al. (2022a)

environment. The second technology is see-through holographic lenses. These allow the user to see the real world nearly without any restrictions and embed the virtual objects into reality by means of holographic projection, based on the measurement of different sensors. In this case, α' is equivalent to α since the user is not seeing a simulacrum of the real world on a display, but the real world itself. An illustration of how this concept refers to a real-world example is visible in Fig. 2.7.

2.2.2.5 Development

For creating AR applications, several development platforms and software development kits (SDKs) are provided. Most of them require significant programming skills and are either commercial or closed-source. Examples include the Unity run-time and development environment, Apples ARKit, Google ARCore, Wikitude, Vuforia,

Kudan, Unreal Engine, or Adobe Aero. In addition, open source platforms and SDKs are available, such as ARToolKit+, OpenXR, or Holokit.

An alternative to the above platforms and SDKs is the WebXR device application programming interface (API) (Jones et al. 2023), which is currently a candidate recommendation for a W3C web standard. It specifies a web API that provides browser-based access to handheld or head-mounted augmented reality and virtual reality devices, including sensors. This allows AR content to be rendered by any compatible WebXR-enabled browser without the need to install additional software or use SDKs. As of today, WebXR is supported, for example, by Chrome and Edge browsers on the Android operating system,¹ including handheld smartphones and tablets, as well as *head-mounted displays*, e.g., the Microsoft HoloLens 2.² Furthermore, WebXR is already included in the WebKit engine used by iOS Safari³ and the recently released visionOS⁴ for the Apple Vision Pro,⁵ but it is not yet activated. A more detailed discussion about the different development platforms for augmented reality applications can be found in Sect. 5.2.1.

2.2.2.6 Conceptual View on Augmented Reality

Looking at augmented reality on a more abstract level, there are different aspects to consider. In the following, we introduce a framework that describes the aspects that need to be considered based on the properties of augmented reality applications, introduced by Muff and Fill (2021b). For all parts of the framework, we distinguish between the *form* that is required to convey information and the *substance* of information transmitted through the *form*. We further consider three aspects that we deem relevant for augmented reality: *context*, *content*, and *interaction*. Figure 2.8 shows the components of the framework and its interconnections.

Context From a *context* point of view, augmented reality applications integrate virtual representations into the context of the real world. This requires the *form* of the context and its substance to be recognized. To consider the *context*, augmented reality applications can monitor the environment through cameras and other sensors and analyze where the user is located, which objects are currently in the user's visible space, and which properties these objects have, i.e., the *form* of the *context*. The augmented reality application can then infer not only the existence of objects but may also classify them further based on additionally perceived attributes, e.g., the current state of a machine or the actions of a person. To this end, it is necessary to

¹ <https://caniuse.com/webxr> last visited on: 01.03.2024.

² <https://microsoft.com/en-us/hololens> last visited on: 01.03.2024.

³ <https://github.com/WebKit/WebKit/tree/main/Source/WebCore/Modules/webxr> last visited on: 01.03.2024.

⁴ <https://developer.apple.com/visionos/> last visited on: 01.03.2024.

⁵ <https://www.apple.com/apple-vision-pro/> last visited on: 01.03.2024.

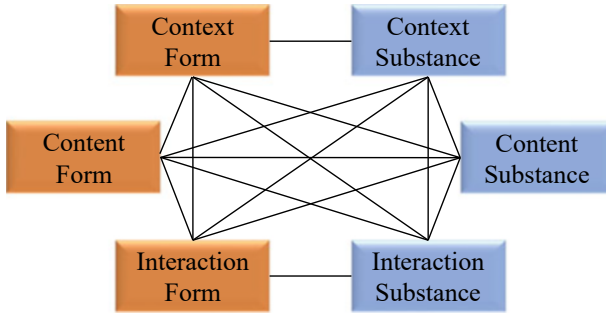


Fig. 2.8 Concept of form, and interaction. Adapted from Muff and Fill (2021b)

interpret perceived attributes and incorporate them into previously stored knowledge to obtain the *context substance* (Muff and Fill 2021b).

Content The *content* aspect focuses on the information that is presented to the user through the augmented reality application. Again, we distinguish between the form of the *content*, i.e., the kind of visual representation that is used to transport the information, e.g., an image, a diagram, or a 3D representation—and the substance of the *content*, i.e., the nature of the information, e.g., numeric data, procedures, textual information, etc. The *content* may thereby be dynamically created or static. To create the form of the *content*, it needs to be reverted to the field of computer graphics where the technical specification of graphical objects and their display on the augmented reality hardware is investigated. Due to recent developments in this area, it can today be chosen from a wide range of APIs that greatly ease the specification of such graphical objects. Regarding the substance of the *content*, it must be decided what is required in the current *context* and how it can be tailored to the needs of a user (Muff and Fill 2021b).

Interaction As showed in the previous section, the *interaction* in augmented reality environments can be done in various ways. Again, we distinguish between the form of *interaction*—e.g., via deviceless *interaction* through gesture recognition or by using devices such as bats for pointing at and selecting objects in 3D environments, and the substance of *interaction*—e.g., the intention of the user and the goal of the *interaction* such as moving an object, selecting an object, entering information, etc. (Muff and Fill 2021b).

Illustration Example By returning to the different dimensions of *context*, *content* and *interaction*, as well as *form* and *substance*, we can analyze the concrete manifestations of these dimensions in a fictional use case (Muff and Fill 2021b). Consider a scenario where a landlord wants to rent out an apartment. She activates her AR equipment because she is uncertain about the legal requirements. The device scans the environment and embeds a legal visualization in the room. The legal visualization shows the process of calculating rent for different types of buildings or facilities, taking into account the legal system. For the *context*, the form of

the *context* is the rental of an apartment and a room in this apartment is used to visualize the legal regulations. The substance of the *context* is here a room, not only the room itself, but also all the objects and properties of this room. This can be information such as the object's wall or ceiling. Additionally, it may be inferred by the [AR](#) application that this room belongs to an apartment and that this apartment is available for rent. The form of the *content* is, in this case, a [3D](#)-model of a legal visualization. Therefore, a model of the legal visualization is projected onto a [3D](#) plane. The nature of the information, i.e., the substance of the *content*, is a mixture of textual and procedural information since the visualization is a kind of process. This should enable the viewer to correctly classify and interpret the given information. The *form of interaction* in the [AR](#) environment solely uses the hands of the user, i.e., it is device-less. Therefore, the virtual object can be placed in the environment by grabbing and dragging the object. The substance of the *interaction*, i.e., the goal of the *interaction* is to place the virtual object in the room in such a way that it is in a useful position for the user to observe it without obstructing or disturbing him (Muff and Fill [2021b](#)).

2.2.3 Metaverse

The term *metaverse* is used in a variety of areas, and at the time of writing this work there is no universally accepted definition of the term. In the understanding of the author of this book, the term metaverse includes all aspects of the reality-virtuality continuum introduced at the beginning of Sect. [2.2](#).

The metaverse is about changing the way humans experience the digital world and the shift from [2D](#) digital experiences to [3D](#) experiences. In this process, our digital lives will increasingly include immersive media that appear all around us and are experienced in the first person. Either on traditional [2D](#) screens as we know them today from smartphones and tablets, or in the future more on [HMDs](#), digital enhanced contact lenses, or through direct projection into the eye.

It will affect everything from how we work, shop and learn online, to how we socialize and organize ourselves. The metaverse is the transition of the digital world from flat content to immersive experiences. Thereby, fully immersive experiences like [VR](#), or virtual extension of the real world, i.e., [AR](#) are equally involved.

Since [VR](#) and [AR](#) technology is still quite expensive and not affordable for everyone today, this process will be strongly directed by technological advances in future years. It was a similar story with smartphone technology. Smartphones are now firmly anchored in society, but were not that popular at first. It took a few years for the technology and the use cases and the possibilities of this technology to evolve to the point where it was convenient for everyone to use a smartphone.

This is expected to be similar for [XR](#) technologies. They will need to evolve over several years and be driven by early adopters before the technology reaches a point where everyone wants an [XR](#) device to experience the metaverse.

2.2.4 Positioning of Extended Reality, Virtual Reality and Augmented Reality

In the context of this work, extended reality technologies, i.e. VR and AR technologies, are examined in relation to metamodeling. In the following chapters, some aspects are examined in their entirety for VR and AR, and some aspects are looked at more closely from the perspective of augmented reality only. This can be justified as follows.

As stated before, virtual reality centers on providing users with fully immersive environments. On the other hand, in augmented reality, the real world is still visible to the user. All relevant aspects, such as 3D visualizations, user positioning, and orientation in VR, are also important for AR. However, not all aspects of AR are also necessary for VR. For example, the mapping between objects in the real world and the virtual environment, or object recognition.

From an objective perspective, VR can be considered a subset of AR. Therefore, at some points in this work, we will refer only to AR, although the fundamental principles are generally applicable to VR as well, thus for XR in general. Furthermore, there is a movement to shift the naming of XR to “*spatial computing*”, especially in relation to the recently released *Apple Vision Pro* device. In this context, *spatial computing* can be seen to be similar to extended reality. *Spatial computing* explicitly includes the interaction with the extended and the real environment, while in XR this is implicitly included. Thus, in the remainder of this work, we will not distinguish further between *extended reality* and *spatial computing*.

2.3 Literature Study of Pairing Conceptual Modeling with VR/AR

Some parts of the content of this section have been published in a similar form as a research paper at the *13th International Symposium, BMSD 2023* with the title: ***Past Achievements and Future Opportunities in Combining Conceptual Modeling with VR/AR: A Systematic Derivation*** (Muff and Fill 2023d). The study was conducted as a separate, self-contained project. Therefore, the methodology is discussed separately again in this section for clarity.

Throughout the last years, the application of extended reality technologies to business scenarios has been increasingly studied by the research community (de Souza Cardoso et al. 2020). As explained in Sect. 2.2, in VR, the user’s perception is entirely based on *virtual* information in a *virtual* world, resulting in a high level of immersion. In augmented reality, computer-generated information is provided to the user in addition to data collected from real life through sensors of the AR device, enhancing the user’s perception of *reality*. Due to recent technological progress (Yin et al. 2021), mobile VR and AR devices became widely available

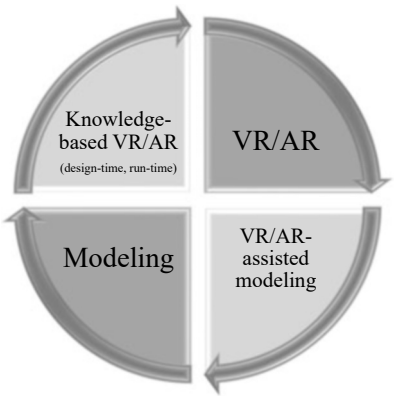
and affordable and allowed the broad application of these technologies in industrial scenarios such as maintenance tasks or training (Grambow et al. 2021).

The development of such applications requires considerable technical know-how. As described by Wild et al. (2020), the provision of *systematic* and at the same time *flexible* approaches for *designing VR* and *AR* applications is considered a prerequisite for a more widespread adoption. Conceptual modeling, for example, as used in enterprise modeling, may serve as a solution for both aspects, since, according to a vision by Sandkuhl et al. (2018), modeling should be part of the everyday work in our future lives—see Chap. 1. On the one hand, conceptual modeling aims to reduce complexity by structuring a particular domain to improve human understanding (Mylopoulos 1992; Cabot and Vallecillo 2022)—cf. Sect. 2.1.1. This may involve the use of novel technologies, for example, in 3D space (Betz et al. 2008). On the other hand, the knowledge made explicit in such models may be processed algorithmically, for example, as found in model-driven engineering to ease the creation of software applications (Brambilla et al. 2017) or to fuel knowledge in existing applications (Fill et al. 2021).

This leads us to propose two main directions for virtual and augmented reality in relation to conceptual modeling. First, the use of the functionalities of *VR* and *AR* for modeling itself. We will denote this as *VR/AR-assisted modeling*. Second, incorporation of information from the model space into *VR* or *AR* applications, which we will denote as *knowledge-based VR/AR*. This second direction includes both design-time and run-time aspects, that is, the modeling and model-driven generation of *VR/AR* applications as well as the fueling of the model content into existing *VR/AR* applications—see Fig. 2.9.

This section aims to explore the multitude of approaches proposed in academic research for combining conceptual modeling with virtual and augmented reality. Despite numerous contributions, to the best of our knowledge, no structured analysis of them has been done so far. Therefore, in this section, we discuss a systematic review of the literature on the combination of conceptual modeling with *VR* and *AR* within the last two decades. Furthermore, we show the process of conducting a

Fig. 2.9 Main directions for virtual and augmented reality in relation to conceptual modeling



computational content analysis to identify distinct research streams that have been explored in this field, and we analyze and refine the results of our analysis with the help of expert classification. The purpose of this study is to provide a comprehensive understanding of the main contributions to combining conceptual modeling with virtual and augmented reality, identify the main topics that have been studied in the past, and highlight areas that require further research.

Since this analysis considers contributions over a time span of two decades, the terms *augmented reality* and *virtual reality* changed during this period. This is the reason why the analysis also contains work about early 3D environments which are not regarded as typical VR or AR environments today. Nevertheless, they are important to get an overview of the research area and to find relevant research streams in later periods.

The remainder of the section is structured as follows. In Sect. 2.3.1, we will describe the research methodology used for the review. Section 2.3.2 will describe the literature search results, which were used as input for latent dirichlet allocation (LDA) to computationally derive a first set of topics. Furthermore, it will be shown how these topics have been refined using expert classification and the allocation of papers to the final set of topics. Finally, we will discuss the results of the analysis in Sect. 2.3.4, including related studies in Sect. 2.3.5 and its limitations in Sect. 2.3.6.

2.3.1 Methodology of the Analysis

The methodology we followed in this study is mainly based on the high-level recommendations by Kitchenham (2004) for conducting systematic literature reviews. This includes the three phases *Planning*, *Conducting*, and *Reporting*. The planning phase includes the identification of the need of the review as described above and the definition of a research protocol; see Fig. 2.10. The research protocol describes each step of the review process according to Booth et al. (2016). For the conduction phase, we further reverted to Webster and Watson (2002), who describe in particular the screening of dedicated outlets and the application of forward- and backward searches. In addition, we performed a computational literature analysis followed by an expert classification to derive the topics of the different research streams.

2.3.1.1 Aims and Scope of the Analysis

The aim of this section is to identify the main research topics that combine conceptual modeling with virtual and augmented reality. Furthermore, the study should provide detailed information on the proposed concepts of *VR/AR-assisted modeling* and *knowledge-based VR/AR*, as described above. The time frame investigated includes academic papers published between 2000 and the first half of 2022, with the aim of showing the most recent research developments in these areas.

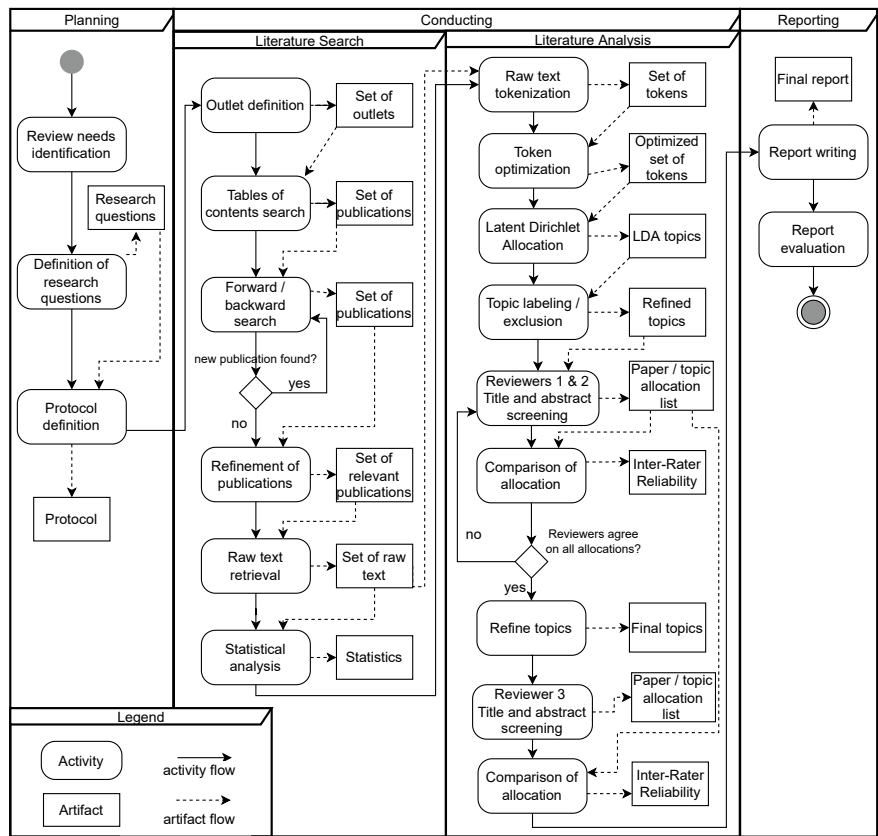


Fig. 2.10 Description of the research protocol. The protocol is divided into the three main areas as proposed by Kitchenham (2004). The process shows the undertaken steps together with the resulting artifacts. Adapted from Muff and Fill (2023d)

2.3.1.2 Methodology of Literature Collection

To identify the main research contributions on the combination of conceptual modeling with VR/AR, we reverted primarily to the method proposed by Webster and Watson (2002) to determine an initial set of relevant scientific publications. In the following, the steps as shown in the *Literature Search* section of the research protocol in Fig. 2.10, are described. We first identified the top nine outlets in conceptual modeling based on a similar study focused on a different community, by Härer and Fill (2020). According to this source, many topics in conceptual modeling are strongly related to enterprise modeling. For example, business/business process models, or data models and schemas. Furthermore, we added six outlets in the area of *Business Informatics* and *Information Systems* in which we assumed potentially relevant contributions in the area of interest (*Outlet definition*)—see step 1 in

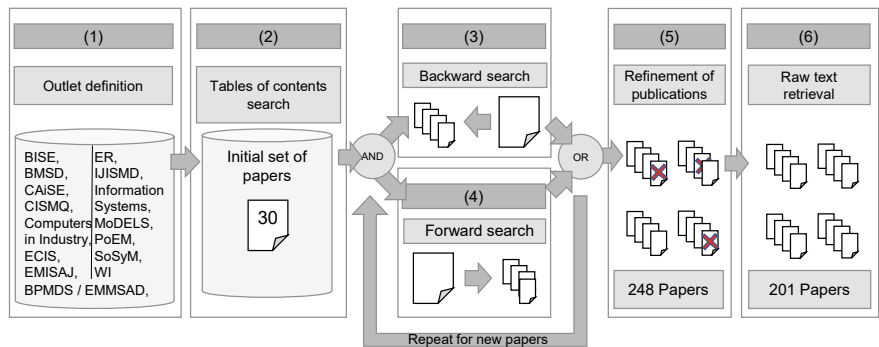


Fig. 2.11 Data collection process according to Webster and Watson (2002): (1) Identification of relevant outlets in the area of interest. (2) Table of contents screening of the relevant outlets and journals from (1). (3 and 4) Iterative forward and backward search based on newly added relevant papers from (2) or previous iterations of (3 and 4), resulting in 258 contributions. (5) Selection refinement by a deeper inspection of the selected papers, resulting in 201 relevant papers. Retrieval of the raw texts of the resulting 201 relevant papers (6). Adapted from Muff and Fill (2023d)

Fig. 2.11. The six additionally chosen outlets are *BISE*, *CAiSE*, *Computers in Industry*, *ECIS*, *Information Systems*, and *WI*.

We analyze the table of contents of the outlets to identify relevant contributions (*Tables of contents search*). For each of the contributions found, we applied a forward- and backward search. This means searching for each paper relevant papers that were cited by this contribution and contributions that cite the respective paper. For this task we used *semanticscholar.org* and *google.scholar.com* (*Forward/backward search*). We repeated this step until we did not find any more new papers; see the loop in *Literature Search* area in Fig. 2.10 after the third activity. We then reviewed the set of papers for excluding wrongly selected papers (*Refinement of publications*). Finally, we retrieved the raw texts of the articles for further analysis (*Raw text retrieval*). Furthermore, we calculated quantitative indicators of the set of relevant papers (*Statistical analysis*).

2.3.1.3 Methodology for Content-Based Data Analysis

To derive the contribution of this research in terms of topics previously studied, we performed a computational analysis and complemented it with an expert-driven classification of relevant papers in distinct topical domains. The following steps refer to the *Literature Analysis* section defined in the research protocol in Fig. 2.10.

2.3.1.3.1 Computational Data Analysis

For the compilation of an initial set of topics that describe the main directions in the articles in the literature analysis, we resorted to the *topic modeling* technique. To conduct topic modeling, the raw text of each document had to be tokenized (*Raw text tokenization*). In addition, preliminary tasks such as minimal stemming, stopword filtering, case transformation, synonym replacement, and single character filtering were conducted (*Token optimization*).

On this basis, topic modeling was conducted, which is an established method that has been successfully applied in previous literature reviews (Härer and Fill 2020; Muff et al. 2022b). **LDA** (Blei et al. 2003) and non-negative matrix factorization (**NMF**) (Shahnaz et al. 2006) are two basic methods that have been used for a long time. They are still used regularly today. **NMF** is increasingly used for document collections with greater noise, i.e., text that cannot be properly categorized using text mining approaches. **LDA** (especially the MALLET implementation (McCallum 2002)) can struggle with noise, but can be used in an iterative and semi-supervised way to produce a good ground truth of topics (Churchill and Singh 2022). When the ground assumption of non-correlating topics does not hold, alternatives such as correlated topic model (**CTM**) and structural topic model (**STM**) can be used. **CTM** is an extension of **LDA** that relaxes the assumption of independent topics (Blei and Lafferty 2005). **STM** is a mixture model in which each document can belong to a mixture of the specified k topics (Roberts et al. 2014). This method is often used for documents that contain questionnaire data with open-ended questions. For datasets consisting mainly of short texts, such as posts on social networks, specific methods have been developed, among others, self-aggregation-based topic model (**SATM**) (Quan et al. 2015), or embedding-based topic model (**ETM**) (Qiang et al. 2017).

In order to choose the right topic modeling method, it is not only the data set that matters, but also the goal one is pursuing with the analysis. The dataset considered for this work consists exclusively of scientific papers, so we could exclude recent methods for short texts. Since we assumed that the topics in our analysis should be unique and independent and we wanted to achieve the clearest possible assignment of a paper to a topic, we could also exclude **CTM** and **STM** as possible models. For the named reasons and since several empirical studies have validated **LDA**'s capability of extracting semantically meaningful topics from texts and categorize texts according to these topics (Boyd-Graber et al. 2014; Chang et al. 2009; Lau et al. 2014; Mimno et al. 2011), we chose traditional **LDA** as our basic methodology. We used MALLET (**MAchine Learning for Language Toolkit**), as well as the **LDA** implementation that is part of RapidMiner Studio 9.5.⁶

LDA works at the level of documents to classify their topics. Compared to simpler approaches such as word frequency, n -gram analysis, and term frequency inverse document frequency (**TF-IDF**), **LDA** constructs a probabilistic model, in

⁶ <https://rapidminer.com/> last visited on: 17.07.2023.

which several topics are considered per document. Over a set of documents, each document d is represented by a statistical distribution θ_d over its different topics. That means that each topic has a certain probability or weight for d , and for each topic k a distribution of words $\theta_{d,k}$ (Blei 2012). The hidden variables of the distributions are computed with the Gibbs sampling scheme by using parallel processing, where the weights per word are determined to maximize their probability of occurring in a given topic (Newman et al. 2009).

For the LDA, we used an iterative approach that tries to optimize the hyperparameters for the topic generation, i.e., the number of topics, alpha and beta heuristics, as well as some evaluation measures like the topic coherence and the topic perplexity (McCallum 2002). Such an iterative approach tries to optimize these parameters in multiple iterations to produce a good result. Evaluating the results of an LDA is difficult, since topic discovery is an unsupervised process. There is no gold standard for evaluating topics. Thus, to evaluate the latent space of topic models, we need to collect exogenous data. Chang et al. (2009) and Mimno et al. (2011) showed that the quality of topics can be measured and compared by the coherence value of the topics. Coherence measures the degree of semantic similarity between high-scoring words in a topic. These measures help to distinguish between topics that are semantically interpretable and topics that are artifacts of statistical inference (Mimno et al. 2011). The goal of our analysis was to obtain distinguishable topics that are semantically coherent and, therefore, human-interpretable.

Another often-used metric for the quality of topic models is the perplexity of models. Perplexity is a statistical measure of how well a probability model predicts a given sample. Chang et al. (2009) showed that human judgment and perplexity often do not correlate, or even anticorrelate. Regarding the optimal topic size, according to Mimno et al. (2011), there is no definitive optimal topic size. However, smaller topics seem to be of better quality.

Taking this information into account, we performed different iterations of LDA and compared the corresponding average coherence values C_{UMass} to decide on the optimal topic size for our analysis. The details and results of this process will be described in Sect. 2.3.2.

2.3.1.3.2 Expert Analysis and Refinement

The topics proposed by the LDA were then manually labeled and refined by the authors of Muff and Fill (2023d) and an external expert in an iterative procedure. By looking at the different words allocated by the LDA to the topics and considering the list of the most probable assigned topic for each paper, we allocated labels to each topic (*Topic labeling/exclusion*). After this first manual topic labeling, the papers were assigned manually to one of the topics. As proposed by Vessey et al. (2002), by screening the titles of the papers, two experts assigned the papers independently of each other to exactly one topic. Each disagreement between the reviewers was then discussed in multiple iterations to find a consensus based on the abstracts of the contributions (*Title and abstract screening*).

To verify the agreement of the reviewers, we calculated the inter-rater reliability (IRR) by using *Cohen's Kappa* (κ) (Cohen 1960) (*Comparison of allocation*). These steps were repeated until reviewers one and two reached an agreement on their allocation. Thereby, the topics could also be refined by renaming them or merging similar topics, if found necessary, during the manual evaluation (*Refine topics*). This resulted in the final list of topics.

As an extension of the labeling process for two reviewers proposed by Vessey et al. (2002), a third reviewer manually assigned the articles to the final topics derived by reviewers one and two through a title and abstract screening (*Reviewer 3 Title and abstract screening*). The goal was to validate the reliability of the final assignment of the first two reviewers. Again, the IRR between the decision of the third reviewer and the joint assignment of reviewers one and two was calculated (*Comparison of allocation*). First, the third reviewer only had the titles of the articles available. In a second iteration, reviewer three looked at the abstracts of the papers to which he did not assign the same topics as reviewers one and two. He then decided whether to assign a different topic.

2.3.2 Results of the Literature Study

In this section we, describe the results obtained from the literature search process (Sect. 2.3.2.1), as well as of the content-based data analysis process described in Sect. 2.3.1.

2.3.2.1 Results of Literature Search

As described in the methodology section above, initially 15 outlets or journals were examined. We manually looked through the content tables of the outlets and searched for the terms *augmented reality*, *virtual reality*, *AR*, *VR*, and *3D*. The abstracts of the resulting set of articles were used to decide whether they are relevant for the analysis. A paper was considered relevant if it addressed at least one of the above areas, as well as conceptual modeling. In the context of Muff and Fill (2023d), conceptual modeling was regarded in a broad sense, that is, relating to the formal description of some aspect of the world around us based on a schema for the purpose of human understanding and communication (Mylopoulos 1992; Härer and Fill 2020). Initial screening of the 15 outlets and journals resulted in a list of 30 relevant initial articles. The forward- and backward searches based on these articles resulted in a list of 248 articles. Subsequently, a more detailed analysis of whether each article indeed involved conceptual modeling, as delimited above, was performed. By manually reviewing the abstracts or full texts as appropriate, the set of papers was reduced to a final list of 201 relevant articles. The lists used for the entire process of this study are available in the online appendix (Muff and Fill 2023a).

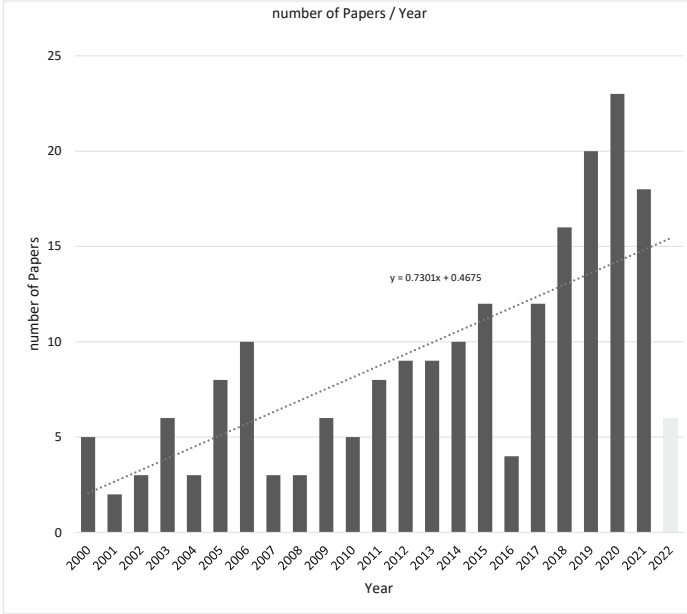


Fig. 2.12 Number of articles published per year with a linear trend line. The year 2022 was not considered since not all publications were yet available at the cut-off date of the analysis. Adapted from Muff and Fill (2023d)

Looking at the number of publications over time, there is a clear upward trend in the number of published papers with a slope of $m = 0.4675$ —see Fig. 2.12. Papers from 2022 were excluded here, since at the time of the analysis not all papers from 2022 were published.

Furthermore, the publications are spread over many different outlets. Only 30 of the 201 relevant papers were published in one of the 15 outlets initially defined. In total, the 201 papers were published in 143 different outlets. 30 of them had more than two, and only 12 of these outlets had three or more publications in the observed time span—see Fig. 2.13. From the initial 15 outlets only *CAiSE*, *BMSD*, *ECIS* and *Computers in Industry* have three or more relevant publications.

2.3.2.2 Results of Computational Topics Analysis

Based on the methodological information for the computational data analysis above, we performed multiple LDAs on our raw data set with seven to 13 topics. Average coherence values C_{UMass} varied between -3.369 and -4.257 , where lower values are considered as better (Mimno et al. 2011)—see Fig. 2.14. Since C_{UMass} decreases rapidly at first and remains relatively stable between the LDAs with ten and 13 topics, we decided to analyze the model with ten topics that have

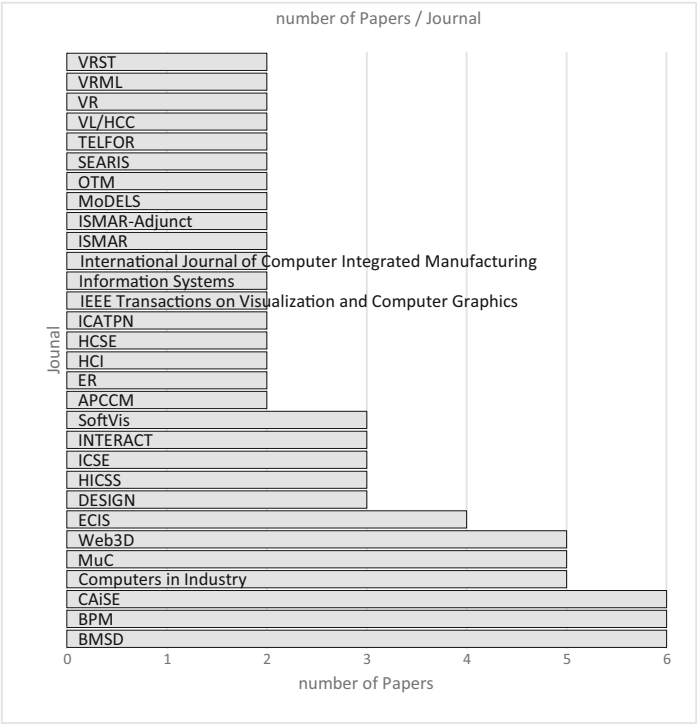


Fig. 2.13 Outlets with two or more relevant papers resulting from the literature search. Adapted from Muff and Fill (2023d)

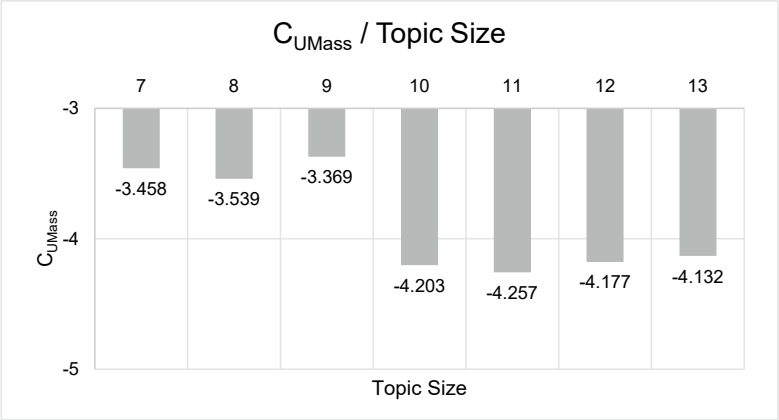


Fig. 2.14 Average coherence values C_{UMass} for the different LDAs for topic sizes between seven to 13. A smaller value is considered as better

an average coherence value of $C_{UMass} = -4.203$. We assume that the coherence would be much better with a very high number of topics. Since subjective labeling of topics would become very difficult with a high number of topics, we limited the number of topics to ten. Furthermore, we chose five tokens per topic as topic size. The results of the LDA with ten topics are visible in Table 2.1. Therefore, for each topic, the five most weighted tokens and their respective weight are listed. For example, *Topic 0* has the most weighted terms *system*, *maintenance*, *context*, *user*, and *information*. The order of the topics does not have a specific meaning. We did not look at the summed weights of the topics since this was not of interest in this analysis.

In addition, the LDA provides a list of all papers with the probability assigned according to the different topics. Over a set of documents, each document d is represented by a statistical distribution θ_d over its different topics. This means that each topic has a certain probability or weight for d , and for each topic k a distribution of words $\theta_{d,k}$ (Blei 2012). The hidden variables of the distributions were computed with the Gibbs sampling scheme using parallel processing, where the weights per word are determined to maximize their probability of occurring in a given topic (Newman et al. 2009).

Only 27 (13%) papers had a most probable assignment of <0.5 to one of the ten topics. The rest of the 174 papers had the most probable assignments of ≥ 0.5 and 101 papers (50%) had the most probable assignments of ≥ 0.7 . This means that the majority of contributions were assigned to one of the ten topics with a much higher probability than to the remaining topics, which can be regarded as a clear allocation to one topic.

Since there is almost no human interference, LDA is a relatively objective process. The LDA process is only influenced by the subjective selection of the analysis parameter, e.g., the topic size. The computational analysis itself is entirely objective and is therefore well suited as an objective ground truth for further analysis. Nevertheless, these first results require some interpretation and contextualization to increase their value.

2.3.3 Result of Topic Refinement

In this section, we show the results of the labeling and revision of the ten initial topics through expert assessment, as well as the allocation of the different papers to these topics.

2.3.3.1 Refined Topics

To label the ten topics, the two authors of Muff and Fill (2023d) considered the words assigned to the topics by the LDA together with the list of the most probable topic for each article, as visible in the research protocol in the *Literature Analysis*

Table 2.1 LDA topics for all papers from 2000 to 2022 with the five most weighted tokens and the according token weights. Topics are not ordered according to summed weights

Topic 0		Topic 1		Topic 2		Topic 3		Topic 4	
Word	Weight	Word	Weight	Word	Weight	Word	Weight	Word	Weight
System	1371	User	1549	Object	2126	Business	956	Content	1409
Maintenance	1248	Application	1312	behavior	644	device	673	semantic	1144
Context	854	Object	1228	Class	627	Task	643	Ontology	697
User	822	Scene	1041	Concept	537	Data	605	Domain	644
Information	785	Interface	1035	Language	466	bpmn	602	Representation	587
	5080		6165		4400		3479		4481
Topic 5		Topic 6		Topic 7		Topic 8		Topic 9	
Word	Weight	Word	Weight	Word	Weight	Word	Weight	Word	Weight
Train	216	Environment	1377	System	1370	System	842	Visualization	1834
Skill	128	Participant	1363	Product	1044	Service	690	Software	1488
Student	111	World	1347	Environment	909	Glass	652	Diagram	1294
Simulation	111	User	1186	sysml	577	Smart	616	uml	1152
Scenario	103	Task	1091	Use	568	Information	597	System	775
	669		6364		4468		3397		6543

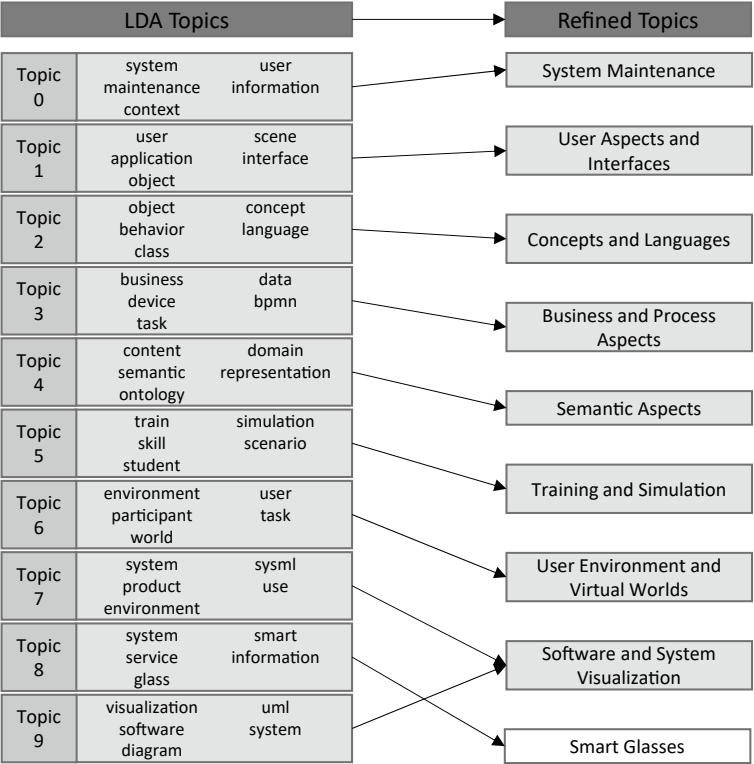


Fig. 2.15 Visualization of the topic evolution in the first refinement steps. **LDA Topics:** Initial topics delivered by the LDA analysis with the five most weighted words each. The order of the topics has no systematic ranking. **Refined Topics:** Topics according to the expert topic labeling

section in Fig. 2.10. As described above, the most probable topic for each article is the one to which the LDA assigned the article with the highest probability.

Then, a label was commonly decided for each LDA topic—see Fig. 2.15. Some topics required specific treatment: *Topic 8* consists of the terms *system*, *service*, *glass*, *smart*, and *information*. This indicates a focus on smart glasses, which have been explicitly researched in several of the selected articles. Since this seems to be hardware-specific, it was decided to exclude this topic from the subsequent analysis.

In the next step, further topic refinements were made. *Topic 7* and *Topic 9* were considered similar in terms of their research area. The terms *sysml*, *uml*, *diagram*, and *visualization* were interpreted as related to software or system visualization. Thus, they were merged into one topic with the label *Software and System Visualization*. As shown in Fig. 2.15 (Refined Topics), of the 10 LDA topics, eight topics were kept for further analysis.

Table 2.2 Agreement measures for Cohen’s Kappa according to Landis and Koch (1977)

Kappa statistic	Strength of the agreement
<0.00	Poor
0.00–0.20	Slight
0.21–0.20	Fair
0.41–0.60	Moderate
0.61–0.80	Substantial
0.81–1.00	Almost perfect

2.3.3.2 Paper Allocation and Final Topics

After the initial labeling of the topics, each article was manually assigned to one of the topics by the two authors of Muff and Fill (2023d) to express the main focus of each paper through a single assignment. The resulting IRR in the form of *Cohen’s Kappa* (Cohen 1960) after the first allocation was $\kappa = 0.617$. According to Landis and Koch (1977) values between 0.6 and 0.8 indicate substantial agreement—see Table 2.2.

After agreeing on the allocation of papers to the various topics, reviewers one and two again discussed and refined the topics. Thereby, the topics *User Aspects and Interfaces*, and *User Environment and Virtual Worlds* were combined into one topic entitled *User Aspects and Development Approaches*, which was considered as a more suitable, common label when inspecting the underlying papers. This resulted in the final set of the seven topics visible in Fig. 2.16 (Final Topics).

As shown in Table 2.3, 63 papers (31.3%) were allocated to the topic *Business and Process Aspects*, followed by 37 papers (18.4%) allocated to *Software and System Visualization*, 31 papers to *User Aspects and Development Approaches* (15.4%), 26 papers to *Semantic Aspects* (12.9%), 23 papers to *Training and Simulation* (11.4%), 14 papers to *Concepts and Languages* (7%), and 7 papers to *System Maintenance* (3.5%). We will discuss the final topics and its main contributions in more detail in the next section.

2.3.3.3 Quality Audit

For additional quality assurance, it was reverted to a third reviewer who assigned the 201 articles to the final seven topics considering only the title of the contributions. The resulting IRR compared to the final allocation of reviewers one and two was $\kappa = 0.520$, indicating moderate agreement (Landis and Koch 1977)—see Table 2.2. After the initial screening of titles, the third reviewer reviewed the abstracts of papers that had not been assigned the same topic as the first two reviewers. The assignments of the other reviewers were not revealed to the third reviewer. Based on the abstracts, the third reviewer decided whether to assign a different topic or maintain the initial selection. After this step, the resulting IRR in comparison to

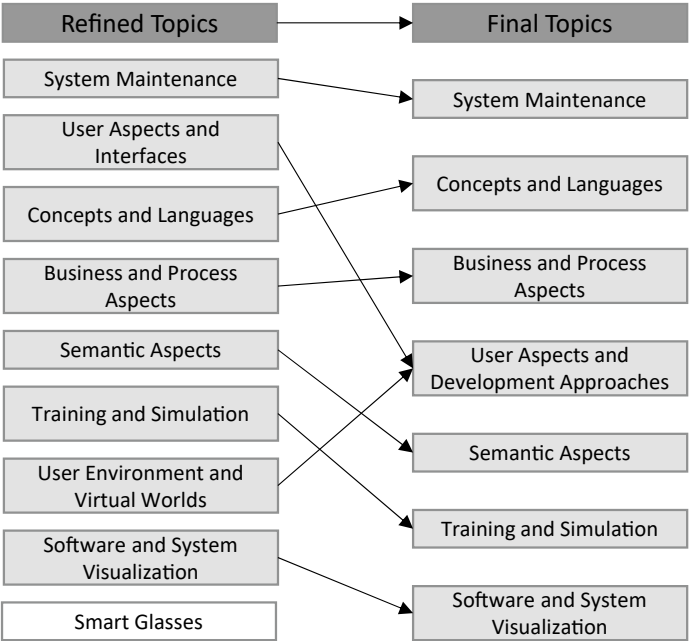


Fig. 2.16 Visualization of the topic evolution in the second refinement steps. **Refined Topics:** Topics according to the expert topic labeling. **Final Topics:** Final seven topics after the last refinement step

Table 2.3 Distribution of the 201 papers (nPapers) over the final seven topics in alphabetical order after the final allocation by reviewers one and two, and a visual distribution of the papers over time. Adapted from Muff and Fill (2023d)

Topic name	nPapers	Dist. 2000–2022
Business and process aspects	63	
Concepts and languages	14	
Semantic aspects	26	
Software and system visualization	37	
System Maintenance	7	
Training and Simulation	23	
User Aspects and Development Approaches	31	
Total	201	

the final allocation of reviewers one and two increased to $\kappa = 0.655$, indicating substantial agreement (Landis and Koch 1977).

With the insights gained above, we can now advance to the discussion of our findings in relation to the main contributions of combining conceptual modeling with virtual and augmented reality, as well as the initial directions proposed for *VR/AR-assisted modeling* and *knowledge-based VR/AR* (see Fig. 2.9). In addition, we will highlight areas that have not yet been covered by research.

2.3.4 Discussion of Findings

This section discusses the findings introduced in the results section above—see Sect. 2.3.2. First, a short overview of the retrieved contributions is given (Sect. 2.3.4.1). Second, the main research streams of the past 20 years in the area are discussed (Sect. 2.3.4.2).

2.3.4.1 Overview of the Retrieved Contributions

The question of the main contributions of combining conceptual modeling with virtual and augmented reality can be answered directly in terms of the literature search (Sect. 2.3.2.1). The 201 relevant publications are spread over several outlets, and no single outlet dominates. The research field under analysis shows a significant increase in publications, suggesting potential for future research efforts.

2.3.4.2 Main Topics of Combining Conceptual Modeling with VR/AR

By discussing the results of Sects. 2.3.2 and 2.3.3, and reflecting on possible application areas that would drive research and industry forward, we can identify the main topics that have been studied in the past and highlight the areas that require further research. Regarding the identification of the main topics, we need to consider the final topics, their interpretation, the allocation of the articles to these topics by the reviewers, as well as some exemplary contributions to these topics. It should be noted that the labeling of the different topics is a subjective task and that other reviewers may assign different labels. However, we tried to mitigate this subjective factor by conducting an objective LDA analysis as a basis for further investigation. Furthermore, the labeling of the different topics was performed by two reviewers in an iterative process, and dissenting opinions were discussed. The final topics and their interpretation, as well as some sample papers that the reviewers assigned to these topics, are discussed in the following.

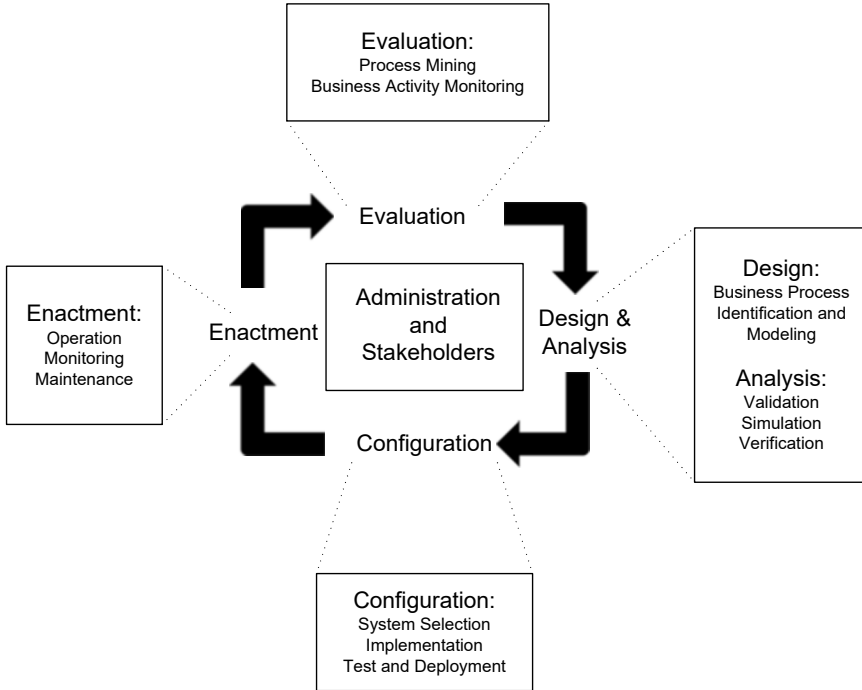


Fig. 2.17 Business process life cycle adapted from Weske (2019)

2.3.4.2.1 Business and Process Aspects

The articles assigned to the topic *Business and Process Aspects* deal mainly with business process management. Regarding the traditional life cycle of business processes according to Weske (2019) (see Fig. 2.17), three of the four components of the life cycle are subject of research related to **VR/AR**. Design/Analysis [R90, R169, R244],⁷ Configuration [R72, R193], and Enactment [R152, R173, R219] have been subject of research related to **VR/AR**. We could not yet discover research on the *Evaluation* of business processes related to **VR/AR**. This is surprising since **VR/AR** devices provide a variety of sensor data that would be predestined for process evaluation. The areas *VR/AR-assisted modeling*, e.g., [R96] and *knowledge-based VR/AR*, e.g., [R1, R158] are both present in research.

⁷ For print readers: All “R” references point to a repository on <https://fabian-muff.github.io/BMSD23>. The files can be downloaded at <https://doi.org/10.5281/zenodo.7794278>.

2.3.4.2.2 Concepts and Languages

The topic *Concepts and Languages* contain contributions like languages for modeling VR/AR systems, or for authoring VR/AR content. Thereby, we could identify the three main streams: *content creation* [R88, R186], *metamodeling* [R147, R26], and *concepts for model-driven code generation* [R184, R119]. All these research streams can be related to *knowledge-based VR/AR*, either for design-time, or for run-time, i.e., real-time content creation. What seems to have not been covered so far is the combination of *knowledge-based VR/AR* and *VR/AR-assisted modeling* in a generic way, e.g., for allowing VR-based model-driven engineering of VR/AR applications, which could be useful for simulating the interaction with 3D environments in VR prior to their realization using AR.

2.3.4.2.3 Semantic Aspects

For structuring the papers allocated to the topic of *Semantic Aspects*, we found that they can be related to the seven components of the semantic web framework derived in Garcia-Castro et al. (2008)—see Table 2.4. Considering these components, we found approaches for *Querying and Reasoning* [R4, R148, R206], *Ontology Engineering* [R41, R205], *Ontology Instance Generation* [R160, R208], and *Semantic Web Services* [R188]. The assignment to *VR/AR-assisted modeling* or *knowledge-based VR/AR* is not always clear. It depends on whether the semantic aspects are used for modeling ontology-driven VR/AR applications [R41], for semantic aspects such as reasoning for AR during run-time [R148], or for generating models by analyzing the sensor data of VR/AR devices. This last point seems to be missing so far in the found papers.

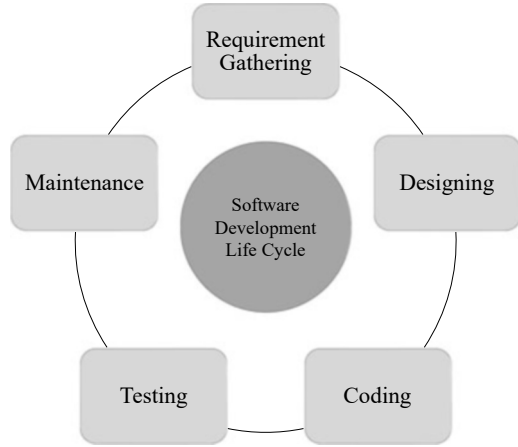
2.3.4.2.4 Software and System Visualization

In *Software and System Visualization* the focus is on *requirement gathering and analysis, designing, coding, testing, and maintenance and support*, that is, on the software development life cycle from Khan (2021)—see Fig. 2.18. Most of the

Table 2.4 Components of the semantic web framework according to Garcia-Castro et al. (2008) with contributions found for the different components of the framework

Semantic component:	Examples:
Data & Metadata management	–
Querying and reasoning	[R4, R148, R206]
Ontology engineering	[R41, R205]
Ontology customization	–
Ontology evolution	–
Ontology instance generation	[R160, R208]
Semantic web services	[R188]

Fig. 2.18 Software development life cycle according to Khan (2021)



discovered papers deal with analyzing [R58, R142, R155, R156 R157] (*knowledge-based VR/AR*) and designing [R105, R177] (*VR/AR-assisted modeling*) software and systems. Only few addressed testing and maintenance of software and systems [R9, R85] and none have addressed so far the coding phase.

2.3.4.2.5 System Maintenance

System Maintenance is an area where VR/AR is used in relation to maintenance activities, e.g., modeling languages and VR/AR systems guiding maintenance processes on the basis of conceptual models [R78, R99]. This refers mainly to the area of *knowledge-based VR/AR* as described at the beginning of the chapter. When considering the different types of maintenance, e.g., *improving*, *preventing*, and *correcting* (Mobley 2011)—see Fig. 2.19, all types are covered by the found approaches, since most of them are not bound to a particular maintenance type.

2.3.4.2.6 Training and Simulation

The contributions in this topic mainly focus on training and simulation aspects, such as business process training. Mostly, contributions in this area can be allocated to the area of *knowledge-based VR/AR for design- or run-time*. Most research is conducted in training applications involving virtual worlds for desktop applications [R8, R34, R121] followed by VR training environments [R182, R234]. Very little research has been done in the area of AR training applications combined with conceptual modeling [R75, R228]. This is an area that should be explored further, as AR training offers many potential application scenarios in different areas, for example, in the education of machine use or in health-related processes.

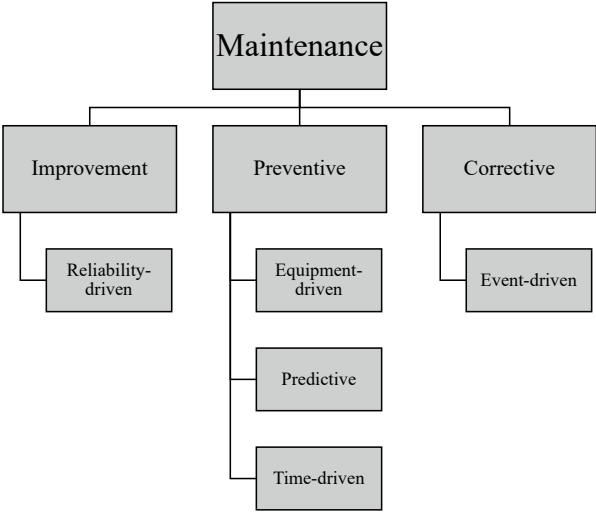


Fig. 2.19 Structure of maintenance according to Mobley (2011)

2.3.4.2.7 User Aspects and Development Approaches

The topic *User Aspects and Development Approaches* is twofold. First, contributions that focus on the user, that is, *user interaction* [R57], *user interfaces* [R29], and *collaboration* [R215]. Second, research that focuses on development approaches, i.e., approaches investigating *content authoring* [R42, R102], *model-driven development* [R30, R46], and the development of *virtual worlds* [R25]. Both of these main streams cover primarily design-time aspects, and therefore, belong to *knowledge-based VR/AR*. Only very few contributions dealt with pedagogic or learning aspects [R132]. This is surprising, as there is a lot of ongoing research on general VR/AR learning approaches, as recently shown by Chen et al. (2017).

2.3.4.2.8 Areas Not Covered in Research

From the above descriptions and the papers mentioned, it becomes clear that most of the contributions found in our analysis are positioned in the area of *knowledge-based VR/AR* where models are used as input for VR/AR applications. Currently, there exist very few approaches where modeling in VR/AR, or the automated elicitation of models is considered. Furthermore, only some contributions focus on the pedagogical and learning aspects in AR modeling. Regarding missing areas, some aspects are not yet covered by research at all. For example, approaches combining knowledge-based VR/AR and VR/AR-assisted modeling, allowing the interplay of these two areas. Further, we could not yet identify approaches on the evaluation of business processes using VR/AR and no approaches for the semantic

elicitation of conceptual models during run-time, e.g., for generating conceptual models based on the user context.

2.3.5 *Related Studies*

Through research and analysis, we state that to this day no literature review has systematically explored the union of conceptual modeling with VR and AR. It is worth mentioning that Pöhler and Teuteberg (2021) conducted a prior review in the field. It is important to note that their focus was specifically on the application of VR for business processes rather than conceptual modeling in its entirety. Thus, our findings highlight the novelty and importance of our review in filling this gap in the existing body of literature.

2.3.6 *Summarized Findings of Literature Study*

In this chapter, we discuss a systematic literature review, a computational bibliometric study, as well as an expert-driven classification of articles that combine conceptual modeling with VR/AR. The analysis indicates a definite upward trend in the number of publications within this research domain. There have been no specific venues for this area so far, but contributions are dispersed across various outlets.

Furthermore, in comparison to the most promising industry use cases as proposed by the Augmented Reality for Enterprise Alliance (AREA) (AREA 2022), which acts under the umbrella of the OMG, 11 out of those 13 use case areas are also covered by our analysis. Only the areas *remote assistance* and *marketing and sales* did not become apparent in our study. This large overlap illustrates the relevance of the topics researched in academia for industry. Table 2.5 shows a description of the AREA use cases, mapped to the topics derived in this analysis.

2.3.6.1 *Discussion of Aspects of Conceptual Modeling with VR/AR*

Research analyzed in this study includes VR/AR-assisted modeling and knowledge-based VR/AR. The emphasis has been heavily on knowledge-based VR/AR thus far, and there are only few publications discussing VR/AR-assisted modeling.

When looking more deeply at the different aspects of VR/AR-assisted modeling and knowledge-based VR/AR, we can again relate to the aspects of conceptual modeling in general, discussed in Sect. 2.1.1. Again, we can distinguish between *interaction aspects*, i.e., VR/AR-assisted modeling, and *flexibility aspects*, i.e., knowledge-based VR/AR. Thus, the interaction aspects relate mainly to *design-time* creation of conceptual models with the help of virtual or augmented reality. On the other hand, knowledge-based VR/AR can be divided into *design-time* and *run-*

Table 2.5 Description of the *Augmented Reality for Enterprise Alliance* (AREA) use cases (AREA 2022), mapped to the topics derived in this analysis

AREA use cases	AREA use case description	Mapping to derived topics
Assembly	This use case pertains to contextually triggered AR-enhanced instructions for “complex assembly” tasks. There are aspects of this use case category that overlap with guidance, remote assistance, and navigation use cases. It may also include an inspection component	-User Aspects and Development -Training and Simulation
Collaboration	This use case pertains to AR assistance for two or more people, either in proximity to one another or at a distance, who are interacting with digital assets for the purpose of collaborative design, creation, and review	-User Aspects Development
Guidance	This use case pertains to contextually triggered, AR-enhanced, step-by-step instructions for any task	-Business and Process Aspects -Training and Simulation
Inspection	This use case pertains to the use of AR for inspecting the results of processes, products and workplaces for safety and quality measurement or documentation. It helps detect and/or reduce incidence of a wide range of risks	-Business and Process Aspects -Maintenance -Software and System Visualization
Maintenance	This use case pertains to use of AR-assisted systems to diagnose issues and, once issues identified, provide digital assets to the user when performing repair and maintenance tasks. There is some overlap with inspection, remote assistance, and complex assembly use cases	-Maintenance
Navigation	This use case pertains to use of AR-assisted systems to direct a person or people from one location to another without delay or safety risks	-Business and Process Aspects
Remote assistance	This use case pertains to having remote subject matter experts assisting on-site technicians or users (together called “on-site users”) in performance or inspection of any task or process which they would not otherwise be able to complete (on time or according to guidelines). There are aspects of this use case category that overlap with guidance, assembly, and navigation use cases. It may also include an inspection component	
Simulation	This use case pertains to using AR to simulate (within and in interaction with the real world) the insertion or repositioning of things using 3D models. The 3D models can include weather patterns, energy flows, industrial equipment, infrastructure (such as HVAC) or moving objects in a zone or confined space (e.g., factory). These are often complex processes requiring employees to lift or perform a process with an odd shape, including serious games (overlaps with training), or to pack diverse objects within volume (e.g., for shipping). Many simulation use cases overlap with visualization use cases and simulation can be used in skill development (training)use cases	-Training and Simulation

(continued)

Table 2.5 (continued)

AREA use cases	AREA use case description	Mapping to derived topics
Situational awareness	This use case pertains to providing digital assets about current (real-time) and historical data to an AR user in context for informed decision-making. It can be a component of other use cases, including complex assembly, collaboration, guidance, inspection, maintenance, and remote assistance	-Semantic Aspects
Training/Education	This use case pertains to using AR to develop any skills and knowledge or fitness that relate to specific workplace competencies and behaviors. It can be a component of other use cases, including complex assembly, inspection, maintenance, and remote assistance	-Training and Simulation
Marketing and sales	This use case pertains to using AR to enhance the marketing and sales process by improving the interaction and engagement between the product and potential purchaser	
Visualization	This use case pertains to using AR to permit employee visualization of digital assets about people, places, or objects in real-world context. The assets may relate to the status of instruments or processes, infrastructure, or moving objects in a zone or confined space (e.g., factory), used to support acquisition of new skills or decision making. This use case can be a component of and overlap with training, situational awareness, simulation, and other use cases	-Software and System Visualization -Training and Simulation
Virtual user interface	This use case pertains to using AR to provide a user with an interactive interface for the real-time operation and control of connected instruments or machines	-User Aspects and Development Approaches

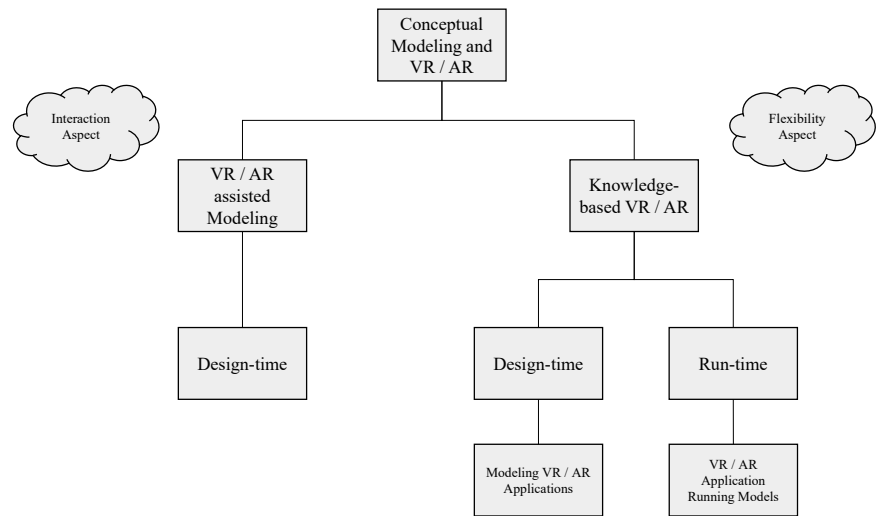


Fig. 2.20 Visualization of the different aspects combining conceptual modeling and **VR/AR** distinguishing interaction aspects and flexibility aspects

time aspects, i.e., modeling of **VR/AR** applications with model-driven approaches, respectively **VR/AR** applications which take conceptual models as input for creating **VR/AR** scenarios. Figure 2.20 shows these different aspects in a visual tree.

2.3.6.2 Limitations of the Literature Study

While we reviewed a substantial number of publications, the study discussed here is not without limitations. To begin with, the selection of literary sources in our research could have been more diverse. Nevertheless, by conducting a comprehensive forward- and backward search for each article, we are confident that we have included the majority of relevant papers. We relied solely on unigrams for our computer-assisted content analysis. We did not consider bi-grams or n-grams, as this would have increased the complexity. This could be considered for future extensions of the study. Finally, we allocated papers to only one topic, following the proposal by Vessey et al. (2002). Nevertheless, multiple allocations could be implemented to gain further understanding of topic overlap.

2.3.6.3 Open Issues

This section presented a valuable overview of the research conducted in the past two decades on the integration of conceptual modeling with virtual and augmented reality, spanning from **3D** environments on **2D** desktops to handheld or head-worn

mobile virtual and augmented reality devices. A variety of different application scenarios in various domains were explored. However, these scenarios were mainly illustrative and did not provide a systematic analysis of the different potential application areas. What is missing is a holistic analysis of the metamodeling area in regard to XR. Therefore, in a first step to such an in depth analysis of pairing conceptual modeling with extended reality, the next chapter will propose a systematic methodology for deriving use cases and offer a comprehensive survey of possible application scenarios, including the derivation of generic requirements for pairing metamodeling with extended reality.

2.3.7 Additional Data of the Analysis

The bibliographies of the document corpora [R1-R248], as well as the various lists [T1-T8] documenting the whole process shown in Fig. 2.10 are available as HTML files online.⁸ In particular, we provide lists with the initial papers [T2], all papers [T3], papers per journal [T4], the most probable topics per paper [T5], as well as the assignments of the reviewers during the review process [T6, T7, T8, T9].

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



⁸ <https://doi.org/10.5281/zenodo.7794278> last visited on: 01.03.2024.

Chapter 3

Derivation of Generic Requirements for Metamodeling for Extended Reality



Some parts of this chapter has been published in a similar form as a research paper in: *12th International Symposium, BMSD 2022* with the title: *Use Cases for Augmented Reality Applications in Enterprise Modeling: A Morphological Analysis* (Muff and Fill 2022b).

This chapter discusses a generic analysis of potential use cases for augmented reality in combination with metamodeling, especially in the area of enterprise modeling. Since the mapping of conceptual knowledge to the real world and vice versa is of particular interest, this analysis mainly focuses on augmented reality and not on virtual reality. As discussed in Chap. 2, the resulting generic requirements are also valid for VR. For the systematic derivation of potential use cases, we refer to *morphological analysis* as a research methodology.

Morphological analysis has a long tradition, and its use as a scientific method is usually attributed to Goethe, who used it to structure organic bodies (Ritchey 2006). However, Zwicky (1989) proposed a modern form of general morphological analysis that can also be applied to abstract phenomena. He describes morphological research as the possibility of seeing and recognizing connections in the totality of material objects, phenomena, ideas, and conceptions, as well as a human activity for constructive creation. The method of morphological analysis comprises the following steps:

1. Exact description and appropriate generalization of the problem.
2. Determination and localization of all parameters determining the solution of the problem.
3. Establishment of the morphological schema from which all solutions of the given problem are inferred without prejudice.
4. Evaluation of all solutions on the basis of a certain chosen standard of values.
5. Choice of the optimal solution and progression to the final design.

In a previous paper, Grum and Gronau (2018) had reverted the methodology of morphological analysis by presenting a morphological schema for bidirectional AR modeling. However, their morphological analysis focuses more on the modeling

activity itself and not on the derivation of new use cases as in this analysis. For the derivation of our morphological schema, expert workshops have been conducted as an additional research methodology (Thoring et al. 2020). In three workshops with five experts, the morphological schema as well as the use cases proposed in this chapter were derived.

3.1 Morphological Schemes for Augmented Reality and Enterprise Modeling

For the derivation of new use cases in the area of enterprise modeling and augmented reality, we designed two morphological schemes—see Figs. 3.1 and 3.2. The morphological schemes represent the two areas of augmented reality and enterprise modeling as origins. These are refined into parameter dimensions. Each of the dimensions has different characteristics, denoted as values. The dimensions and the according values were defined at the beginning of the workshops by analyzing the domain of the two origins.

Origin	Dimension	Values
AR	GPS Position	Yes / No
	Indoor Position	Yes / No
	Relative Position	Yes / No
	Orientation	User-Device / Device Orientation
	Eye Tracking	Yes / No
	Type of Device	HMD / Tablet / Smartphone / Artificial Lenses
	Audio Input	Yes / No
	Audio Output	Yes / No
	Acceleration Data	Yes / No
	Depth Camera	Yes / No
	Camera	Yes / No
	Gesture Recognition	Yes / No
	3D Controllers	Yes / No
	Internet Connectivity	Yes / No
	Collaboration	Yes / No

Fig. 3.1 Morphological schema for deriving use cases in the origin augmented reality

Origin	Dimension	Values
Enterprise Modeling	Perspective	Strategic Perspective / Business Process Perspective / IT Perspective
	Knowledge Elicitation	Formal / Semi-Formal
	Simulation	Yes / No
	Human Understanding	Expert Level / Laymen Level
	Machine Processing	Yes / No
	State	As-Is / To-Be

Fig. 3.2 Morphological schema for use cases in the origin enterprise modeling

3.1.1 Description of the Solution Space

For the origin *augmented reality*, we included the following dimensions, which are commonly found in state-of-the-art *AR* devices: The tracking of the Global Positioning System (*GPS*) position, the tracking of the indoor position, the tracking of the relative position in regard to the user position, the orientation of the device—i.e., whether it is set by the user or inferred from the device, the availability of eye tracking, the type of device, the availability of audio input, output, acceleration data, a depth camera, a standard camera, gesture recognition, *3D* controllers, internet connectivity, and collaboration features.

For the origin enterprise modeling, we first specified the main focus of a solution in terms of the *perspective* (Frank 2014; Sandkuhl et al. 2014). Possible values are the *Strategic Perspective*, e.g., including business models, performance indicator models, capability models, or product models. Second, the *Business Process Perspective*, e.g., comprising business process models, organizational models, or skill models. And finally, the *IT Perspective*, e.g., including IT service models, IT architecture models, or SLA models. In addition, we distinguish between two types of *knowledge elicitation* for representing enterprise models (Bork and Fill 2014; Fraser et al. 1994): Formal and semi-formal representations. Further dimensions that were added specify whether *simulation* of the models is possible, whether the models are directed towards experts or laymen (*human understanding*), whether *machine processing* of the models is possible, and whether the models are of descriptive (as-is) or prescriptive (to-be) *state*—see Fig. 3.2. When combining values across dimensions, there is a vast number of possible combinations. The combination of all possible values shown in Figs. 3.1 and 3.2, would yield a total of 6,291,456 solutions. Since an analysis of such a large number of solutions is not feasible in practice, we introduced limits for the potential solution space.

3.1.2 Limiting the Solution Space

For the origin *augmented reality*, we limited the origin to the use of head-mounted displays. In particular, we consider the characteristics of the Microsoft HoloLens.¹ This excludes by default the use of GPS positions, indoor positions, eye tracking, and 3D controllers, since the device does not support these features—see upper part of Fig. 3.3 for supported features. Thus, these dimensions are held constant, and thereby reduce the number of possible solutions. For the origin *enterprise modeling*, we chose to analyze the values for the *perspective* dimension separately. Thereby, we keep one perspective constant, e.g., the business process perspective, and vary only the other dimensions. This resulted in a reduced solution space, as shown in Fig. 3.3.

With the restriction on AR HoloLens for the first origin and to the three perspectives in the second origin, the total number of possibilities is reduced to 96 combinations. In the following sections, the results of our workshops are described. In the workshops, we analyzed the 96 combinations and derived new use cases for the combination of EM and AR.

3.1.3 Exemplary Use Case for the Strategic Perspective

On the strategic perspective, the workshop provided us with multiple use cases. For the first expert workshop, the *Perspective* dimension has been fixed to the value *Strategic Perspective*. This perspective looks at models like business models, performance indicator models, capability models, or product models. In the following, we describe a use case for the collaborative modeling of a business model canvas (BMC) (Osterwalder and Pigneur 2010; Wieland and Fill 2020). The BMC is used to support the design of business models in a visual way. Since there is no formal definition of BMCs, the dimension of *Knowledge Elicitation* is set to *Semi-Formal*. There is no *Simulation* available for the classical BMC² and the *Human Understanding* dimension is more oriented towards laymen rather than experts in respect to the required EM skills. A traditional BMC is not machine-processable. However, since the processing of the model is required for a use case like the one described in the following, we revert here to an extension of BMC that allows interconnection of different constructs of the BMC and makes it, therefore, processable (Wieland and Fill 2020). Thus, the dimension *Machine Processing* is set to *Yes*. Further, a BMC is mostly about a desirable state in the future, which is why the value of the *State* dimension is *To-Be*. The values of the different dimensions for the strategic perspective are marked in Fig. 3.4 with blue rhombuses (◆).

¹ <https://www.microsoft.com/en-us/hololens> last visited on: 01.03.2024.

² Extensions allowing for simulation have been proposed by Romero et al. (2015).

Origin	Dimension	Values			
AR	GPS Position	Yes	No	●	▲
	Indoor Position	Yes	No	●	▲
	Relative Position	Yes	●	▲	
	Orientation	User / Device	●	▲	Device Orientation
	Eye Tracking	Yes	No	●	▲
	Type of Device	HMD	●	▲	Tablet
	Audio Input	Yes	●	▲	No
	Audio Output	Yes	●	▲	No
	Acceleration Data	Yes	●	▲	No
	Depth Camera	Yes	●	▲	No
	Camera	Yes	●	▲	No
	Gesture Recognition	Yes	●	▲	No
	3D Controllers	Yes	●	▲	No
	Internet Connectivity	Yes	●	▲	No
Enterprise Modeling	Collaboration	Yes	●	▲	No
	Level	Strategic Level	●	▲	IT Level
	Knowledge Elicitation	Formal	●	▲	Semi-Formal
	Simulation	Yes	●	▲	No
	Human Understanding	Expert Level	●	▲	Laymen Level
	Machine Processing	Yes	●	▲	No
	State	As-Is	●	▲	To-Be
			●	▲	
			●	▲	
			●	▲	
		Smartphone	Artificial Lenses		

Fig. 3.3 Morphological schema for the different perspective dimensions: strategic perspective (◆), business process perspective (●), and IT perspective (▲). Adapted from Muff and Fill (2022b)

Origin	Dimension	Values	
AR HoloLens	GPS Position	Yes	No ♦
	Indoor Position	Yes	No ♦
	Relative Position	Yes	♦
	Orientation	User / Device	Device Orientation
	Eye Tracking	Yes	No ♦
	Type of Device	HMD	Tablet
	Audio Input	Yes	No
	Audio Output	Yes	No
	Acceleration Data	Yes	No
	Depth Camera	Yes	No
	Camera	Yes	No
	Gesture Recognition	Yes	No
	3D Controllers	Yes	No ♦
	Internet Connectivity	Yes	No
Enterprise Modeling	Collaboration	Yes	No
	Perspective	Strategic Perspective ♦	Business Process Perspective
	Knowledge Elicitation	Formal	Semi-Formal ♦
	Simulation	Yes	No ♦
	Human Understanding	Expert Level	Laymen Level ♦
	Machine Processing	Yes	No
	State	As-Is	To-Be ♦

Smartphone

Artificial Lenses

IT Perspective

Fig. 3.4 Morphological schema for the use case on the strategic perspective (♦)

When looking at the [AR](#) HoloLens origin, we must exclude the dimensions of *GPS Position*, *Indoor Positioning*, *Eye Tracking* and *3D Controller*, since the device does not support such features. In terms of collaboration, multiple users have access to the same model, i.e., the modeling canvas—dimension of *Collaboration*. When multiple users collaborate remotely, representative avatars can be visualized in the local [AR](#) environment in relative position to the model. This helps overcome barriers related to location-independent collaboration. Since the canvas could be virtually projected using an [AR](#) device, the participants can be at different locations for remote modeling and analysis. The model may be rendered in different sizes and the participants could walk around the *3D* model. Therefore, the morphological schema is set to *Relative Positioning* and *User/Device* orientation. To retrieve information about objects in the real environment and the movement of the user, *Acceleration Data*, *Depth Cameras* and normal *Cameras* are used in the [HMD](#).

When it comes to the modeling part, i.e., interaction with the models, various options are made available. Any part of the model could be manipulated through voice commands or hand gestures—dimensions of *Gesture Recognition* and *Audio Input*. One could navigate through the model and modify parts of the model with a voice assistant. Further, there may be a modeling assistant that guides the user through the modeling process similar to a voice bot—dimensions of *Audio Input/Output*. In addition to voice and gesture control, real objects could be linked to the model. Camera sensors may be used, e.g., to digitize documents or notes and integrate them directly into the model—again the dimension of *Camera*.

When looking at the information available for modeling, the user and the assistant have access to various other enterprise models and IT systems to retrieve additional information required to create the [BMC](#). For example, there could be visual or acoustic information on financial indicators related to the business model. By analyzing sensor data and other information sources, states and consequences can be inferred through ontological reasoning. Since this requires additional external information, we revert to the dimension of *Internet Connectivity*.

Depending on the technical background of the participants, some information can be more or less important. Thus, the different views of the same model components can be visualized in different levels of detail for the individual user. Furthermore, if based on metamodeling, all models created in [AR](#) can be visualized and modified in a traditional *2D* or *3D* modeling environment and vice versa.

From a general perspective, this use case would create a multitude of new opportunities. On the one hand, modeling could become easier and more intuitive, and guided modeling would allow non-experts to participate in the modeling process. On the other hand, the visual connection to other areas of business modeling makes it easier to perceive important information and relationships. In summary, strategic decisions could be made more easily and based on better information by modeling business model canvases in [AR](#).

Figure 3.5 illustrates an example of an augmented reality application for modeling a business model canvas in a collaborative environment.



Fig. 3.5 Example of the **BMC** modeling use case in augmented reality. Adapted from *Gorodenkoff/stock.adobe.com* and Fill and Muff (2024)

3.1.4 Exemplary Use Case for the Business Process Perspective

The *Business Process Perspective* includes all enterprise modeling activities related to business processes. In the expert workshop sessions, we encountered use cases in the area of process elicitation, process support, and process control. In this section, we present a holistic use case as one example. It is about the elicitation, subsequent execution, and monitoring of a future industrial manufacturing process using BPMN in combination with **AR**. The characteristics of the morphological schema for the business process perspective use case are marked as green circles (●) in Fig. 3.6.

Knowledge elicitation is set to *Semi-Formal* as this is common for *BPMN* models. Furthermore, the use case does not concern *Simulation* and the dimension on *Human Understanding* is more on the *Laymen Level* than for experts. The process can be processed by machines—dimension of *Machine Processing*. Since the use case is about the elicitation of a future process, the *State* dimension is set to *To-Be*.

Looking at the **AR** HoloLens origin, we must again exclude the dimensions of *GPS Position*, *Indoor Positioning*, *Eye Tracking* and *3D Controller*. In the use case at hand, the process already exists in the manufacturing process but is not yet standardized.

To capture the process, a technical expert uses the **AR** device, by which the physical location of the user and the activity are captured relative to an initial marker pattern provided by the user. The marker defines the origin of the application, and all positions are calculated relative to this origin. This relates to the dimension of *Relative Positioning* and to *User/Device* orientation and *Acceleration Data*. Furthermore, the **AR** application recognizes that the expert performs some activity

Origin	Dimension	Values		
AR HoloLens	GPS Position	Yes	No	●
	Indoor Position	Yes	No	●
	Relative Position	Yes	●	
	Orientation	User / Device	●	Device Orientation
	Eye Tracking	Yes	No	●
	Type of Device	HMD	●	Tablet
	Audio Input	Yes	●	No
	Audio Output	Yes	●	No
	Acceleration Data	Yes	●	No
	Depth Camera	Yes	●	No
	Camera	Yes	●	No
	Gesture Recognition	Yes	●	No
	3D Controllers	Yes	No	●
	Internet Connectivity	Yes	●	No
	Collaboration	Yes	●	No
Enterprise Modeling	Perspective	Strategic Perspective	●	Business Process Perspective
	Knowledge Elicitation	Formal		Semi-Formal
	Simulation	Yes	No	●
	Human Understanding	Expert Level		Laymen Level
	Machine Processing	Yes	●	No
	State	As-Is		To-Be
				●

Fig. 3.6 Morphological schema for the use case of the business process perspective (●)

with the help of camera sensors. This relates to the dimensions of *Camera* and *Depth Camera*. He can define tasks of a process and label them as desired using a voice assistant and hand gestures—dimension of *Audio Input/Output* and *Gesture Recognition*. In addition, correct sequences, control rules, information flows, and responsibilities can be defined using voice commands. This information is then visualized on the AR device. Furthermore, the work environment is captured by the AR sensors and objects such as machines or environment conditions are added to the model via annotations—again using *Camera* and *Depth Camera*. Through semantic analysis of this information for the respective process step, additional actions or measures can be inferred and suggested to be included in the model—for example, workplace safety measures; see, e.g., Muff and Fill (2022a). For repetitive activities, the expert can store various objects as a 3D model and document the specified movements of the objects by executing the activity. Since these actions may require external services for object recognition through machine learning and semantic reasoning that may not run on the AR device, the device needs an internet connection.

After completion of the interactive process elicitation, a domain expert can check the process model in a conventional 2D environment. The captured process can now be used to support employees during process execution or to ensure the correct execution of the process. There are two options for process support. First, the process can be visualized. In this case, the entire process is shown as a BPMN diagram superimposed on the real environment using the position information relative to the initial marker. If the user follows the process flow, he can walk through it and analyze the entire process in the real environment. The user could further analyze the process collaboratively, where all users see the same information using their own AR device—*Collaboration* dimension.

The second option focuses on real-time assistance during process execution. Thus, the user receives visual or acoustic instructions about the current task. If the user is not already at the right position, visual hints are shown to guide the user to the right position. If a user needs hints about the information flow or responsibilities, he can ask for them by voice commands, and the appropriate information will be visualized—*Audio Input* dimension. In addition, holographic overlays on objects are visualized, assuming that they were defined during the process elicitation, e.g., through video recordings and object recognition. These show the user the exact procedure to perform a specific task. For example, the movement of a wrench is displayed at the specified position in the form of a pre-recorded video. During task execution, additional actions or measures can be inferred and suggested by semantic reasoning.

Since the exact procedure of the process is determined during the process elicitation, the application can also be used to check if everything was done correctly. If the user accidentally skips a task, the AR application notifies him and guides him to the correct task. Figure 3.7 shows an exemplary AR scenario where the user is guided through a manufacturing process by visual cues to the next activity location and a process overview.



Fig. 3.7 Example of augmented reality business process guiding during a manufacturer process. Adapted from *Tierney/stock.adobe.com*

From a business process perspective, this use case presents multiple opportunities to improve process management using [AR](#). Process elicitation can be carried out by process experts who are not trained in traditional process modeling, i.e., at a *Laymen Level* on the *Human Understanding* dimension, with the assistance of a voice assistant and corresponding visual overlays during real-world activities. This can mitigate misunderstandings between domain experts and modeling experts. In addition, the technical connection to other enterprise models can improve the workflow by providing the right information at the right time. Last, cognitive pressure can be reduced in critical processes, since the system monitors the activities of the user with the help of the marker-based relative positioning and informs him in critical situations.

3.1.5 Exemplary Use Case for the IT Perspective

The last [EM](#) perspective is the *IT Perspective*. In the context of this work, this contains all enterprise models concerning IT-services and IT-infrastructure. For this use case, we analyze a scenario in which an organization's hardware landscape will be represented through the use of *ArchiMate* models. The characteristics according to the morphological schema for this use case are marked as red triangles (▲) in [Fig. 3.8](#).

Origin	Dimension	Values		
AR HoloLens	GPS Position	Yes	No	▲
	Indoor Position	Yes	No	▲
	Relative Position	Yes	▲	
	Orientation	User / Device	Device Orientation	
	Eye Tracking	Yes	No	▲
	Type of Device	HMD	Tablet	
	Audio Input	Yes	No	
	Audio Output	Yes	No	
	Acceleration Data	Yes	No	
	Depth Camera	Yes	No	
	Camera	Yes	No	
	Gesture Recognition	Yes	No	
	3D Controllers	Yes	No	▲
	Internet Connectivity	Yes	No	
	Collaboration	Yes	No	
	Perspective	Strategic Perspective	Business Process Perspective	IT Perspective ▲
Enterprise Modeling	Knowledge Elicitation	Formal	Semi-Formal	▲
	Simulation	Yes	No	▲
	Human Understanding	Expert Level	Laymen Level	
	Machine Processing	Yes	No	▲
	State	As-Is	To-Be	

Fig. 3.8 Morphological schema for the use case of the IT perspective (▲)

As in the two other perspectives described above, the dimension of *Knowledge Elicitation* for the *IT Perspective* is *Semi-Formal*. For this use case, we can distinguish between the model elicitation and the use of the already finished model. In both cases there is no *Simulation* and the dimension of *Human Understanding* is set to *Expert Level*, since the modeling user must have a good understanding of IT-architecture modeling. In the use case there is no *Machine Processing*. Since the IT-architecture is already established, the *State* is now descriptive. Consequently, the dimension value is assigned as *As-Is*.

With respect to *AR HoloLens* origin, we must exclude the same dimensions as in the two use cases above. For the model elicitation, the user of the *AR* application must go to the physical location where the hardware infrastructure is located. For example, the user can go to the server room of the enterprise, where all the servers for the internal applications are running. When starting the application, the virtual world is mapped to the real world by means of *Relative Positioning* and the user's movements are tracked with *User/Device* orientation and *Acceleration Data*. When approaching a server rack, the devices sensors capture the server rack and all available information about it, including unique markers for identifying a particular rack: Dimension of *depth camera* and normal *Camera*. The user can select IT services running on the server from a list by making hand gestures or using voice commands—dimensions of *Audio Input* and *Gesture Recognition*. If the application has not been saved in the model, the user can use a voice assistant to generate a new software service element, i.e., *Audio Input/Output*. The application and related information, as well as the person responsible for providing that information on the IT service layer in the enterprise model, are specified. The user is able to determine the various relationships of the object, and all of these relationships are displayed in a holographic image for an overview of the object's dependencies. The enterprise architecture model can then be visualized and modified on a traditional *2D* modeling platform.

If the user revisits the server room of the enterprise, he or she can get an overview of the infrastructure by visualizing selected properties of the model layer. The user can utilize gesture navigation and voice commands to select different views and examine the relationships with other layers—again *Audio Input* and *Gesture Recognition*. Furthermore, sensors of the *AR* device can retrieve information about the environment and with the help of semantic reasoning, additional information can be inferred. Here, we assume that *Internet Connectivity* is required. For instance, the system can present data on heat clusters, and the issue's specific area is visualized via the augmented reality device. If two server racks are located adjacent to each other and they both house essential systems, the application will point out the problem in a visual way to explain the issue. The real-world mapped models, along with *3D* models independent of location, can be viewed with the cooperation of other *AR* users to discuss the model—dimension of *Collaboration*.

Using such a system enables the modeler and maintainer of enterprise models to gain a comprehensive understanding of the system. Visualizing dependencies between individual components in a location-based manner improves comprehension and expedites the identification of problems and risks. Therefore, enterprise

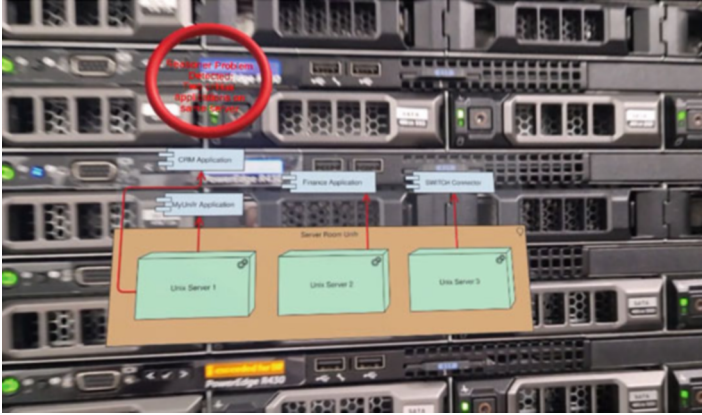


Fig. 3.9 Example showing an augmented reality application for solving server problems in an enterprise server room. The application is reasoning over the context information collected by sensor data and shows the user a warning: “Two critical applications on one server!”

architecture can be optimized in the real environment and problems and risks can be identified more quickly.

Figure 3.9 shows an example of an augmented reality application for solving server problems in an enterprise server room. Thereby the system visually shows dependencies between different servers and dangerous software dependencies based on an *ArchiMate* IT infrastructure model. In this example, the application is reasoning over the context information collected by sensor data and shows the user a warning: “Two critical applications on one server!”

3.2 Resulting Generic Requirements

Based on the three holistic use cases presented, the following global generic requirements (GGR) for metamodeling for extended reality can be derived. These extend the technical requirements for AR-based enterprise modeling that have already been discussed in Muff and Fill (2021a), Muff et al. (2022a) and in Chap. 2. The following requirements are important for the future of AR- and VR-enabled enterprise modeling scenarios and the combination of metamodeling with extended reality in general:

- In addition to hand gestures and voice control, other interaction options adapted for virtual and augmented reality should also be enabled.
- It must be possible to attach virtual objects to real objects by means of anchoring, i.e., reference point information must be stored somehow in the model.
- Object recognition during run-time must be enabled, e.g., by using machine learning algorithms for object recognition, as well as semantic reasoning.

- The accurate positioning of both the user and objects in the physical setting must be determined, including indoor and outdoor environments.
- To achieve an overlay of the real and the virtual objects and to annotate them with additional information, the detection of real objects must be possible.
- Based on the real environment, semantic inferences, i.e., states and consequences about the user's context must be possible. Possibilities to enable this are, e.g., ontological or rule-based reasoning.
- Real-time collaboration should be supported, whether modeling in a multi-user environment in the same location or over a distance.
- Models must support different views of the same model for different situations. This may encompass contextual data, for instance, the level of comprehension of the user.
- The connection of related models and the appropriate visualization of these connections must be enabled.

Since we presented only an excerpt of possible use cases, the list of generic requirements is not complete, and there may be other important areas to work on as well. A more in-depth derivation of specific requirements to join metamodeling with extended reality will be discussed in Chap. 4.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



Chapter 4

Specific Requirements for Metamodeling for Extended Reality



To combine metamodeling with extended reality, different requirements have to be met. This chapter introduces new concepts necessary for the combination of the two domains and derives specific requirements for metamodeling environments considering **VR** and augmented reality (**AR**) in different ways.

Augmented and virtual reality are based on three fundamental core concepts from the field of computer vision (Schmalstieg and Höllerer 2016): (1) *Detectables/Trackables*, (2) *Coordinate Mappings*, and (3) *Augmentations*. In the following, different aspects of these three core concepts, as well as other concepts for three-dimensional environments, such as interaction, or context information, are discussed in more detail. Consequently, specific requirements for the combination of metamodeling with extended reality are derived, and the influence of the generic and specific requirements for metamodeling in combination with extended reality (**XR**) are discussed.

4.1 Methodology for Requirements Derivation

To derive specific requirements for metamodeling environments considering **VR** and **AR**, we analyzed the relevant literature on virtual and augmented reality, focusing on the most important concepts of the technology—see also Sect. 2.2.

Therefore, we analyzed the foundational works of Schmalstieg and Höllerer (2016) and Doerner et al. (2022). This revealed the three mentioned core areas of *Coordinate Mappings*, *Augmentations*, and *Detection and Tracking*. Additionally, the *Interaction* with **VR** and **AR** environment plays an important role.

After analyzing the main concepts of **VR** and **AR**, we evaluated their potential impact on metamodeling in **VR** or **AR** from various perspectives. This includes the different aspects that combine conceptual modeling and **VR/AR**, as introduced in Sect. 2.3.6.1, as well as broader areas of metamodeling in general. The following

sections provide in-depth introductions to various important concepts and outline specific, metamodeling related requirements for all the introduced concept.

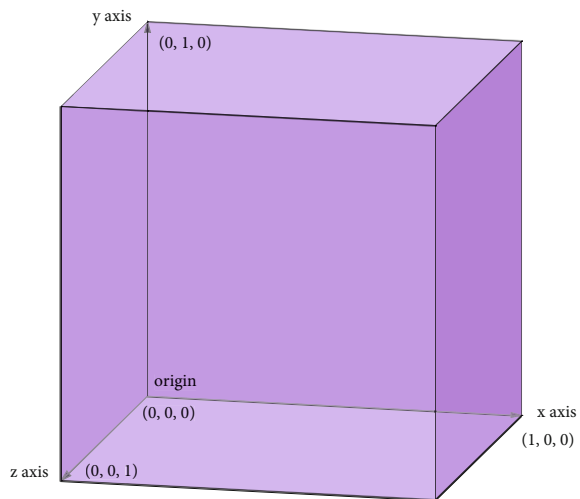
4.2 Three-Dimensional Coordinate Mappings

Three-dimensional coordinate mappings are among the most important concepts in **VR** and **AR**. Every **VR** and **AR** application is based on three-dimensional (**3D**) coordinates and therefore on a **3D** coordinate system with its positions, orientations, and possible transformations. Three-dimensional means, that there are three different axes to consider. The x axis, the y axis, and the z axis. The origin of a coordinate system is always the intersection point of all the axes. In this case, the point $(x = 0, y = 0, z = 0)$, also denoted as $(0, 0, 0)$. This is essential since, in **VR** and **AR**, users can move freely in the environment. Unlike traditional desktop applications where a user mainly uses a mouse to move a pointer on the x and y axes, **VR** and **AR** applications require a third dimension, namely the z axis.

4.2.1 Coordinate Systems

There are different types of **3D** coordinate systems. In the following, we will introduce two particular coordinate systems. As mentioned above, every **3D** space has an origin $(0, 0, 0)$. Figure 4.1 depicts a **3D** base coordinate system with the three axes x , y , and z . The purple cube spans a three-dimensional space of one unit in each direction. Every **VR** and **AR** application is based on exactly one such **3D** base

Fig. 4.1 Example of a **3D** base coordinate system with the three axes x , y , z . The purple cube spans a three-dimensional space of one unit in each direction



coordinate system (Schmalstieg and Höllerer 2016, p. 87), that is always specific for one VR or AR experience.

In addition to the base coordinate system, there are two other important types of coordinate systems to consider. *Relative coordinate systems* and *absolute coordinate systems*.

4.2.1.1 Relative Coordinate System

A three-dimensional coordinate system can have one or multiple embedded coordinate systems, denoted as *relative coordinate system* in the context of this book. Embedded means that the origin (0, 0, 0) corresponds to another point in another coordinate system, i.e., its base coordinate system. Figure 4.2 shows an example of a coordinate system embedded in another coordinate system. This second coordinate system (red cube) has its own origin (0, 0, 0). This origin also has its coordinates relative to the base coordinate system (purple cube).

Every relative coordinate system can again have an infinite number of relative coordinate systems that have their own origin that is relative to its parent coordinate system. Figure 4.3 shows a base coordinate system (purple cube) with an embedded relative coordinate system (red cube) that has again an embedded coordinate system (orange cube).

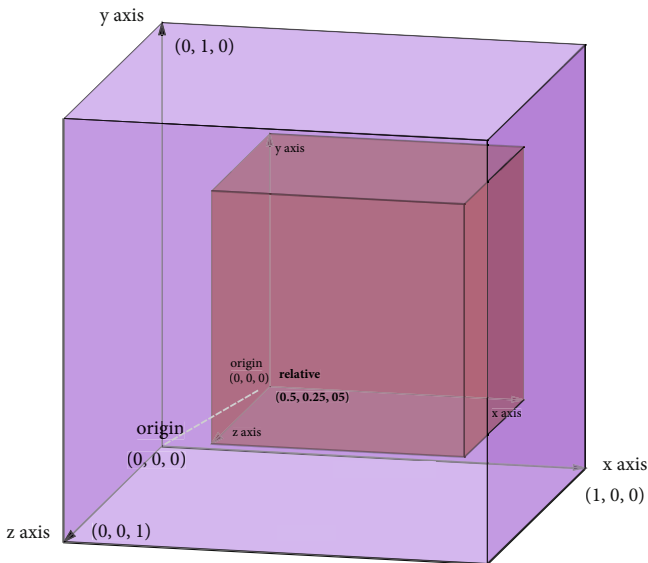


Fig. 4.2 Example of a 3D base coordinate system with the three axes x , y , and z . Inside this 3D base coordinate system there is a second coordinate system (red cube) with its own origin (0, 0, 0) and its relative coordinates to the base coordinate system (purple cube)

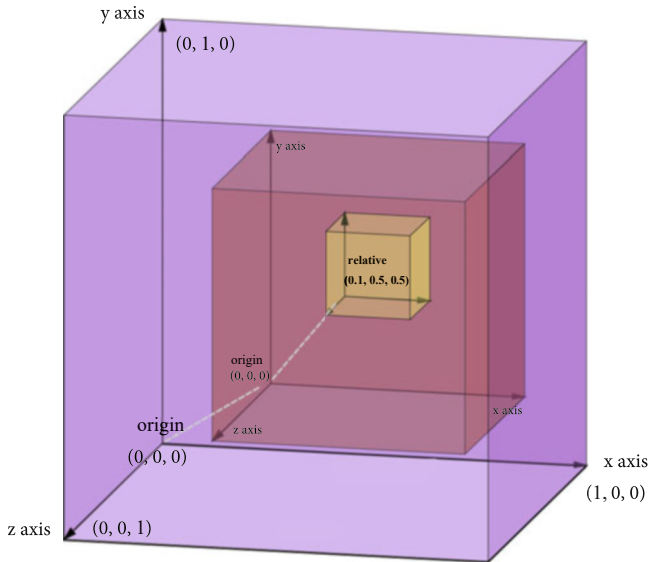


Fig. 4.3 Example of a 3D coordinate system with the three axes x , y , z . Inside this 3D coordinate system there is a second coordinate system (red cube). This relative coordinate system has again relative coordinate system that has its own origin that is relative to its parent coordinate system (orange cube)

This concept is so important because every virtual information that should be embedded into a virtual environment spans its own coordinate system. Thus, every virtual information has its own 3D coordinate system that is a relative coordinate system for its base coordinate system. This base coordinate system can be either the base coordinate system of the virtual environment, or it can be a relative coordinate system of another coordinate system. This plays a very important role in virtual scene management, since these relative coordinate systems build the base for the transformation and rotation of virtual information in the virtual environment.

4.2.1.2 Absolute Coordinate System

Unlike relative coordinate systems, absolute coordinate systems are not based on the base coordinate system of a virtual environment. Instead, they are based on geographical world coordinates, such as latitude, longitude, and altitude.

The equator serves as the reference point for latitude, whereas the Prime Meridian (Greenwich Meridian) serves as the reference point for longitude. Latitude and longitude coordinates are measured in degrees, minutes, and seconds. Altitude is typically expressed in meters. Latitude values range from -90° to 90° , while longitude values range from -180° to 180° . The most widely used geographic coordinate system is the World Geodetic System 1984 (WGS 84) (National Imagery

and Mapping Agency 1991). As an example, the position of the summit of Mount Everest, the highest mountain on Earth, could be denoted as:

$$(\textit{Latitude}, \textit{Longitude}, \textit{Altitude}) = (+27.5916, +086.5640, +8850)$$

One of the most used global navigation satellite systems, the Global Positioning System (GPS) is based on the WGS84 (Gettling 1993). A user's position on the Earth's surface can be calculated by receiving signals from four or more satellites with known orbit positions. The accuracy of this measurement can range from 1 to 100 meters, depending on the number of visible satellites, the quality of the receiver device, and the conditions of signal reception. Altitude, which is more susceptible to measurement errors, is often neglected, and only longitude and latitude are considered (Schmalstieg and Höllerer 2016).

Such global navigation satellite systems allow for the calculation of a device's position by delivering *latitude*, *longitude*, and if necessary, *altitude*. Since the Earth is a sphere, this cannot be considered as normal 3D coordinate system as introduced above. However, it is possible to map a world coordinate system to a 3D coordinate system of a virtual environment. If the real-world coordinates of a VR or AR device are known in relation to the base coordinate system of the virtual environment, it is possible to translate various 3D coordinates into world coordinates and vice versa. Here, the altitude can be provided either by the GPS system or, if the virtual environment is based on the Earth's surface, directly by the virtual environment.

Therefore, especially in an AR environment, it may be useful to use a world coordinate system and the user's compass orientation in combination with the base coordinate system to calculate the relative coordinates of embedded virtual information or real objects. Figure 4.4 depicts the exemplary mapping of a GPS device with a base coordinate system of a virtual environment, for example, from an AR experience. Thus, it is necessary to know the GPS position and compass orientation of both the AR device and the GPS receiver being mapped. The GPS receiver's relative 3D coordinates can be calculated using this information. With the same approach, the relative position of virtual information can be calculated if one knows the real-world position where the virtual information should be visualized in the virtual environment. We will not go into more technical details about the calculation of relative coordinates based on GPS positions and compass orientation at this point.

4.2.2 Position, Orientation and Coordinate Transformations

Every point in a three-dimensional space can be identified by its Cartesian coordinates x , y , and z , which are unique. This uniqueness of the representation of each element is expected from any coordinate system. Therefore, if two coordinate systems are present, it is expected that there is a one-to-one correspondence between

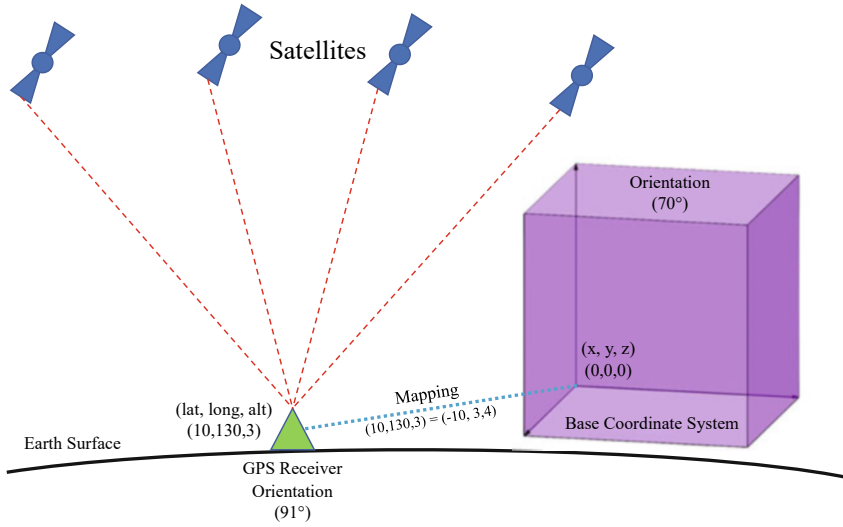


Fig. 4.4 Example of the mapping between a 3D base coordinate system with the coordinates and the rotation of a GPS device

the points in the first system K_1 and the points in the second system K_2 (Karpfinger 2022).

Therefore, in addition to the different coordinate systems, the concepts *position*, *orientation*, *scale*, as well as the transformations between different coordinate systems must be considered.

4.2.2.1 Position

In a 3D coordinate system, a 3D position p is denoted as a vector of three elements $p = (p_x, p_y, p_z)$. This describes a position relative to the x -, y -, and z axis with an offset from the origin of the coordinate system. A position can be noted in the column vector form (Riley et al. 2006):

$$p = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

Thus, p_x , p_y , and p_z are the offset of a point in the direction of the three axes x , y , and z . This can also be described as the position of the origin of one coordinate frame in relation to another coordinate frame with a translation vector

(3×1) (Maxwell 1958):

$$t = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

If we talk of the relation of one coordinate frame to another, this always means that a transformation is applied with regard to the origin of one coordinate system to another coordinate system.

4.2.2.2 Orientation

In addition to position, the orientation in 3D space plays a crucial role for XR experiences. In three-dimensional space, coordinate system rotations of the x -, y -, and z axis in a counterclockwise direction looking toward the origin of the axis can be described with the three matrices R_x , R_y , and R_z (Arfken 1985):

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & \sin \alpha \\ 0 & -\sin \alpha & \cos \alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & -\sin \beta \\ 0 & 1 & 0 \\ \sin \beta & 0 & \cos \beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos \gamma & \sin \gamma & 0 \\ -\sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Thus, the orientation of one coordinate frame with respect to another coordinate frame can be directly described with a 3×3 rotation matrix which is the triple matrix product of R_x , R_y , and R_z (Goldstein 1980, p. 146–147):

$$R = R_x(\alpha) R_y(\beta) R_z(\gamma) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

The rotation matrix itself represents the actual rotation based on a rotation axis and a rotation angle. Rotation around the x -, y -, and z axis has some drawbacks. With rotation matrices, only rotations around the three main axes can be achieved, and the order of rotations around the axes plays a role for the resulting orientation.

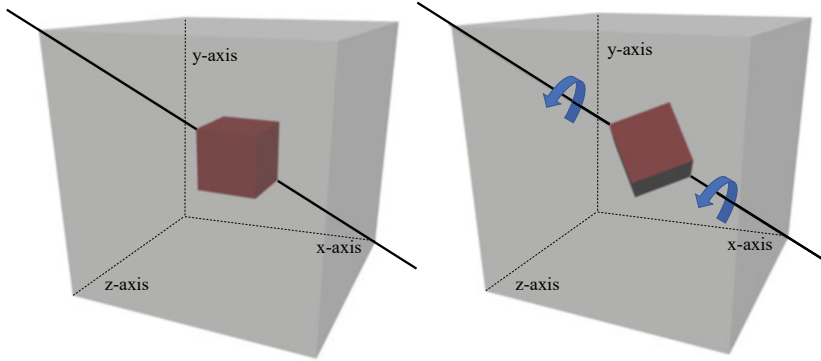


Fig. 4.5 Example of a coordinate frame before (left) and after (right) a quaternion rotation around an exemplary unit vector

One possibility of avoiding these problems is the use of quaternions. Quaternions describe any 3D rotation around an arbitrary axis using only four numbers, instead of the nine typically required with transformation matrices. Generally, quaternions are denoted as in Stephenson (1966):

$$p = a + ib + jc + kd$$

where i , j , and k are so-called unit-vectors and a , b , c , and d are real numbers. Quaternions are thought to be the most effective way of representing the orientation between two coordinate frames due to their compactness, numerical stability, and efficiency, which surpasses that of rotation matrices. Figure 4.5 shows an example of an exemplary coordinate frame that is rotated around an exemplary unit vector.

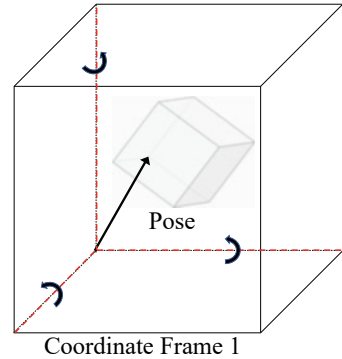
4.2.2.3 Pose

To represent the position and orientation, denoted as pose, of one coordinate frame in relation to another, a 4×4 homogeneous transformation matrix can be used (LaValle 2023):

$$H = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

This is basically the same as combining a rotation matrix and a translation vector—see above. Again, the order of operations is critical. In VR and AR this is important to represent the position and orientation of one coordinate frame in

Fig. 4.6 Example of a *Pose*, i.e., the *Position* and *Orientation* in relation to a base coordinate frame in 3D space. The pose represents a translation in position and orientation of a coordinate frame in relation to another coordinate frame



respect to another. For instance, to indicate the location and orientation of the user within the virtual environment's base coordinate system.

Figure 4.6 shows an example that visualizes the *Pose*, i.e., the *Position* and *Orientation* of a coordinate frame in relation to another coordinate frame.

4.2.3 Requirements of Coordinate Mappings

When considering the requirements for combining virtual and augmented reality with metamodeling, it is necessary to have concepts for all introduced mappings. Therefore, different *global specific requirements* (GSR) emerge.

To ensure clarity, we will refer to an XR-enabled metamodeling environment simply as “a metamodeling environment” when discussing global specific requirements. Thus, a metamodeling environment requires consideration of the local base coordinate system for a VR/AR application for modeling or model execution—see also use cases in Sect. 3.2.

- **GSR1:** A metamodeling environment must support three-dimensional coordinates for the base coordinate system.

A metamodeling environment should consider the concepts of relative coordinate systems for VR/AR applications to enable the positioning and orientation of virtual information in relation to other coordinate systems, whether it is the base coordinate system or the relative coordinate systems of real or virtual objects.

- **GSR2:** A metamodeling environment must support three-dimensional relative coordinates.

A metamodeling environment should consider the concepts of absolute coordinate systems, mainly for AR applications, to enable the mapping of real-world coordinate systems to the base coordinate system.

- **GSR3:** A metamodeling environment must support absolute coordinates.

A metamodeling environment should consider the concepts of 3D positions to allow positioning in three-dimensional space.

- **GSR4:** A metamodeling environment must support 3D coordinates for positioning.

A metamodeling environment should consider the concepts of 3D rotations to allow rotations in three-dimensional space.

- **GSR5:** A metamodeling environment must support 3D rotations.

GSR1 to **GSR5** are very basic and considered as the basis for further requirements in the following sections.

4.3 Visualization of 3D Model Components

Metamodeling can be performed by textual descriptions alone, e.g., by formally expressing metamodels and model instances via mathematical notation as proposed in Fill et al. (2012b). Furthermore, numerous metamodeling platforms enable metamodel definition with visual representations of language concepts. This makes it much easier for non-expert users to create their own modeling language and model instances.

Most traditional modeling platforms only consider the two-dimensional (2D) space and hence the 2D visualizations for model components. Looking at a specific example, ADOxx proposes a generic visual representation language called *GRAPHREP* that allows the generic specification of graphical 2D representations for the different classes and relationclasses (Fill and Karagiannis 2013). Furthermore, it is also possible to define the dynamic behavior of the graphical 2D representation based on information retrieved from the metamodel or model instances. Figure 4.7 shows an example of the ADOxx Development Toolkit application during the definition of a 2D visual representation of the class *Entity* for ER diagrams (Chen 1976). In this regard, it is possible to define not only the static visual representation, but also the dynamic behavior based on information from the metamodel or model instances. For example, the color of an element can be changed at run-time based on an attribute value.

When considering three-dimensional environments, new challenges arise to visualize model components in metamodeling. If a user is capable of navigating in a three-dimensional space, it is reasonable to expect virtual objects in virtual space to also be three-dimensional. In the following sections, we will discuss properties of three-dimensional objects, challenges for dynamic behavior of such 3D objects as well as resulting requirements for metamodeling.

Fig. 4.7 Example of the ADOxx Development Toolkit application during the definition of a 2D visual representation of the class *Entity* for ER diagrams

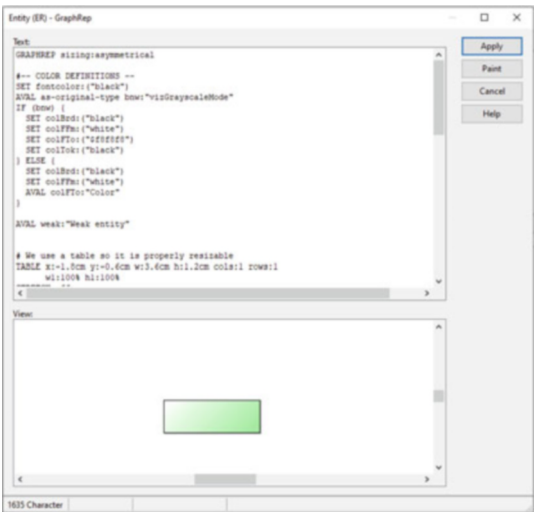


Fig. 4.8 Examples of the use of 2D visualizations in a 2D augmented reality environment. The left image shows a 2D ER diagram in AR. The right image shows a 2D ArchiMate model in AR. Reprint from Hostettler (2022)

4.3.1 Translation of Two-Dimensional Notations

The simplest method to produce 3D visualizations is to translate traditional 2D visualizations to a plane that can be visualized in a 3D environment. Figure 4.8 shows an example of what 2D visualizations could look like when simply translated to a plane in 3D in an AR environment.

Projecting traditional 2D visualizations onto a plane in 3D enables the display of basic or dynamic graphics. This technique again diminishes the added dimension of a 3D environment. This leads to the requirement for the visualization of 3D model components. 3D components are commonly called “3D Geometry”. The modeling of geometries and appearances has a wide range of uses, both in the professional and personal areas. 3D design tools are used to create models for transportation,

architecture, mechanical and electrical engineering, video games, films, and [VR/AR](#) environments (Schmalstieg and Höllerer 2016).

There are two main approaches to describe three-dimensional geometries. Parametric modeling, as they are often used in computer-aided design ([CAD](#)), and polygonal modeling.

4.3.2 Parametric Modeling

Parametric modeling is a method that uses mathematical equations and parameters to create [3D](#) shapes. For example, spheres can be defined by a radius, and cylinders by defining height and diameter. Software such as [SolidWorks](#), and [SketchUp](#) allow users to generate and modify [3D](#) models by adjusting the parameters and restrictions of the shapes, making parametric modeling accurate, adaptable and effective for engineering and design objectives. However, with parametric modeling, it is difficult to create organic or complex shapes that do not follow simple patterns or rules. Thus, in [VR](#) and [AR](#), polygonal modeling is mostly used to create [3D](#) geometries.

4.3.3 Polygonal Modeling

All polygonal geometries are founded on vertices (points), edges (lines), faces, and volumes (Schmalstieg and Höllerer 2016; Paquette 2013). An edge is formed from two vertices in a [3D](#) coordinate system. Several edges can be combined to create faces, and multiple faces can be combined to form closed volumes. Figure 4.9 shows an example of a plane, a cube, and a torus knot in a [3D](#) design tool showing the vertices, edges, and faces of the geometries from a front view perspective. The geometries are constructed using triangular surfaces that are combined to form a

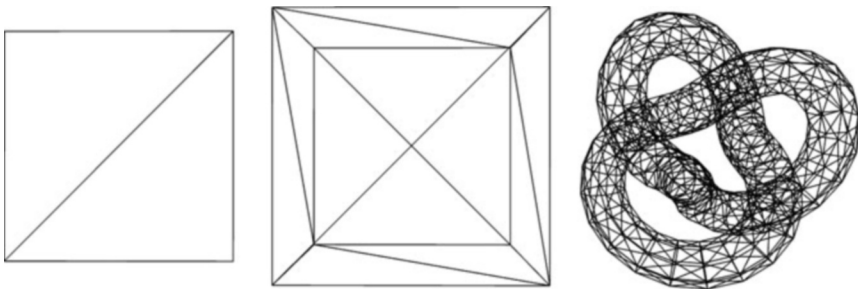


Fig. 4.9 Example of a plane, a cube, and a torus knot in a [3D](#) design tool showing the vertices, edges, and planes of the geometries from a perspective front view

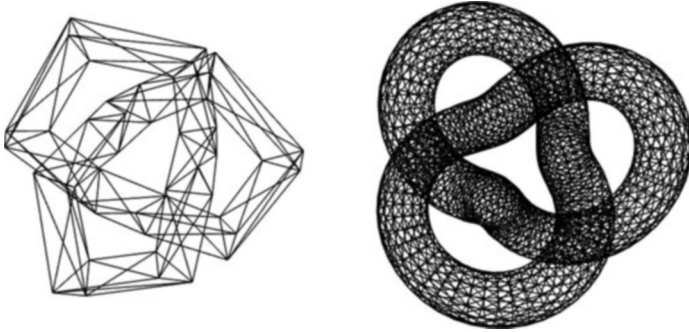


Fig. 4.10 Example of a torus knot geometry, one with only few vertices, edges, and faces (left) in comparison to the same torus knot geometry with much more vertices, edges, and faces (right)

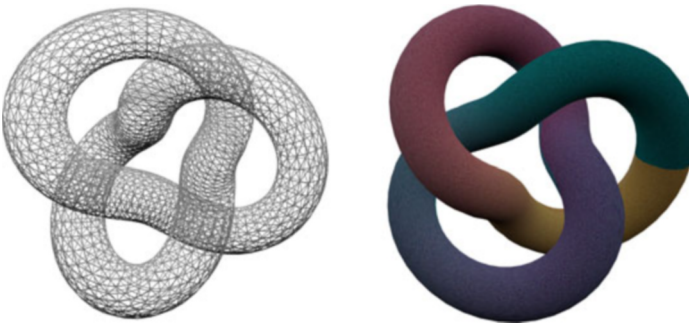


Fig. 4.11 Example of a torus knot geometry without material (left) in comparison to the same torus knot geometry with a material and a image texture with random colors (right)

volume. The size of these triangles can be varied depending on the desired level of detail for the 3D shape.

Figure 4.10 displays the same geometry twice. The first instance shows a representation of the object with limited vertices, edges, and faces. The second instance represents the same object with many more vertices, edges, and faces. As evident in this example, the surface of a geometry appears smoother with an increasing number of triangles, making the polygonal modeling approach more convenient for complex scenarios compared to the parametric modeling approach.

The concept of geometry solely defines an object’s geometric properties. It is important to note that every three-dimensional object can have a material, which can be imagined as a skin wrapped around the triangle mesh of the geometry (Paquette 2013). Such a material can have different properties such as colors, roughness, textures, etc. (Blinn and Newell 1976). Figure 4.11 shows an example of the same geometry, once without a material and once with a material and texture.

4.3.4 Dynamic Behavior of Three-Dimensional Visualizations

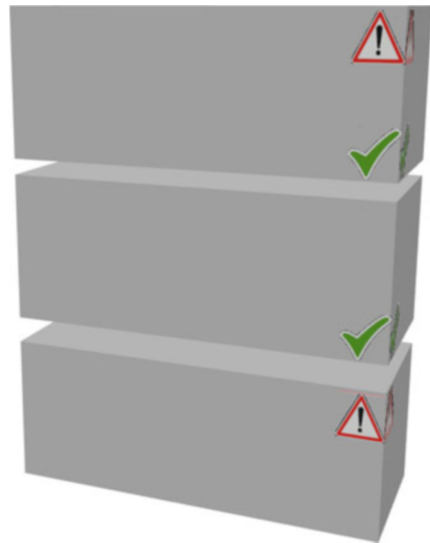
As is the case with two-dimensional visualizations in traditional modeling environments, visualizations in 3D environments must also be able to modify their visualization based on metamodels or model components. There are different possibilities to change the appearance of a 3D object.

In the simplest case, the properties of a geometry, such as its position, visibility, or orientation, can be simply changed. This leads to a directly visible change in the objects visualization. In addition, it is also possible to change properties of the material, such as the color or texture. This also leads to an instant change in appearance.

Figure 4.12 shows an example of dynamically changing the properties of a 3D object. The 3D object on top has three geometries. Below are two instances of the same 3D object, one with the top-right geometry visible and the other with the bottom-right geometry visible. Let us imagine that the top object is the standard visualization of a meta-concept. The two objects below could then be instances of this meta-concept, changing their visualization based on properties of the instance of this meta-concept. For example, a BPMN task could show either a warning or a success sign depending on the message flows entering a task. The different visualization options would have to be defined generically in the metamodel.

Another well-known possibility to change the visualization of 3D objects is the dynamic change of properties in a predefined order, also denoted as *animation*. Animation in computer graphics means change over time (Paquette 2013, p. 239). Time is represented by a sequence of numbered frames. In each frame of an animation, properties of a 3D object may change, e.g., the position of vertices,

Fig. 4.12 Example of changing the properties of a 3D object dynamically. The 3D object on top has three geometries. Below are two instances of the same 3D object, one with the top-right geometry visible and the other with the bottom-right geometry visible



the visibility of whole geometries or the texture of a material. By chaining such frames sequentially, the visualization of 3D objects can be dynamically changed. Such animations are stored on a 3D object and can be played back on demand during a VR/AR experience.

4.3.5 Three-Dimensional Data Formats

To describe 3D geometries, different data formats are available for different tools, and most of them are interchangeable. The Graphics Library Transmission Format (GLTF)¹ is the most used file format, while Apple’s products use the universal scene description (USD)² specification for defining 3D scenes and objects.

As visible in the schematic diagram in Fig. 4.13, GLTF describes a whole 3D scene, i.e., a scene based on a base coordinate system (cf. Sect. 4.2). A GLTF file describes multiple Nodes, which can be Cameras, Skins, or Meshes, where meshes are 3D objects. Additionally, a GLTF file can contain Images that are utilized for Textures, which in turn are utilized for Materials, which ultimately are utilized for

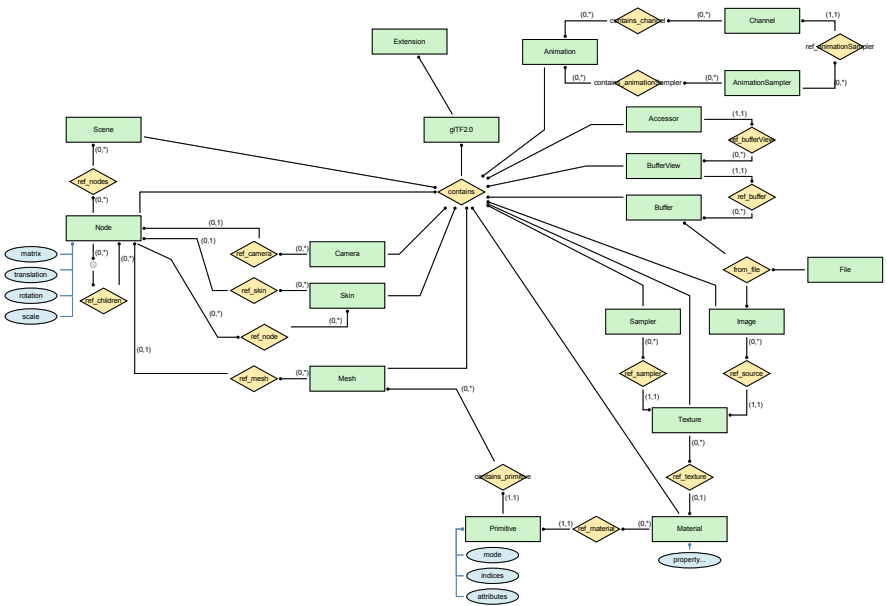


Fig. 4.13 ER diagram of the GLTF data format specification

¹ <https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html> last visited on: 01.03.2024.

² https://openusd.org/release/spec_usdz.html last visited on: 01.03.2024.

Due to the introduction of a third dimension in [XR](#) environments, the visualization of three-dimensional objects must be considered.

- **GSR6:** A metamodeling environment must allow for the visualization of [3D](#) objects.

Since the definition of such three-dimensional objects can be very complex, standardized data formats must be used.

- **GSR7:** A metamodeling environment must allow the use of well-known [3D](#) data formats.

Furthermore, since metamodeling defines visual representations not for each instance, but on a generic level in a metamodel, concepts are needed to define three-dimensional visualizations in a generic way.

- **GSR8:** A metamodeling environment must allow for the definition of [3D](#) visualization on the level of metamodels.

In terms of dynamic behaviors of three-dimensional visualizations, dynamic behavior can be achieved by reacting to changes in the properties of a model instance. This can be done by either changing the properties of the [3D](#) objects or by playing back an animation on a [3D](#) object.

- **GSR9:** A metamodeling environment must allow dynamic changes of three-dimensional visualizations.

Again, since metamodeling defines visual representations not for each instance, but on a generic level in a metamodel, concepts are needed to define the dynamic behavior of three-dimensional visualizations in a generic way.

- **GSR10:** A metamodeling environment must support concepts to define the dynamic behavior of model components on the level of metamodels.

4.4 Detection and Tracking of the Environment and Real-World Objects

Tracking is the process of dynamically determining spatial properties during runtime of a system. In the context of virtual and augmented reality, tracking an object involves continuous measurement of its position and orientation. Various objects can be tracked in extended reality, including the user's head, eyes, or extremities, as well as [XR](#) devices like mobile phones or head-mounted displays, or any object present in the [XR](#) scene (Schmalstieg and Höllerer 2016, p. 85).

The use of a tracking system requires the management of multiple coordinate systems—see Sect. 4.2. In order for virtual objects to be correctly superimposed on tracked physical objects, these coordinate systems must be synchronized (Holloway 1997). This process, called registration, is necessary to convert the tracking poses to the coordinate system of the rendering application (Schmalstieg and Höllerer 2016, p. 179).

To display virtual objects registered to real objects in 3D space, it is necessary to determine the relative pose. This involves acquiring the position and orientation of the AR device in relation to the real objects. Since VR and AR applications operate in real-time, the tracking process is continuous over time (Schmalstieg and Höllerer 2016, p. 87).

The concept of *Detection* is closely related to tracking. It involves not only tracking the real environment, but also detecting specific instances of real objects, such as images, markers, or real-world objects, based on 3D models of these objects. This concept is highly relevant to the field of computer vision, but technical details will not be discussed here. Nevertheless, it is important to note that the detection of real-world objects must occur continuously, and all the tracking features introduced above are also utilized to track the detected objects.

Many of the concepts outlined in the following sections are aligned with the AR Foundation specification,³ which provides a universal definition of essential tracking concepts for augmented reality. The objective is to guarantee platform-independence of the concepts—see Table 4.1. The features of *Session* and *Camera* are very basic technical features, which will not be covered in this chapter.

4.4.1 Device and Object Tracking

There exist different tracking facets in the area of extended reality. In this section, we introduce the most important device and object tracking approaches.

4.4.1.1 Device Tracking

Device tracking is the most basic tracking approach and is necessary for all extended reality experiences. Thereby, the device's position and orientation in relation to the base coordinate system are tracked (Azuma 1993)—see Sect. 4.2. In order to track the user's movements, different sensors, such as camera sensors, accelerometer, or gyroscope, are needed. This is not only necessary for the display device itself, but also for additional input devices, e.g., hand-held controller devices.

4.4.1.2 Image and Object Tracking

The tracking and detection of 2D images can form real-life reference points utilizing predefined reference images. Computer vision algorithms are used to detect these images and estimate their position and orientation relative to the base coordinate system. This calculated pose can then be used to attach virtual objects to the established anchor.

³ <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.0/manual/index.html> last visited on: 01.03.2024.

Table 4.1 Feature list of the AR Foundation specification. Adapted from <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.0/manual/index.html>

Feature	AR foundation description
Session	Enable, disable, and configure AR on the target platform.
Camera	Render images from device cameras and perform light estimation.
Device tracking	Track the device’s position and rotation in physical space.
Image tracking	Detect and track 2D images.
Object tracking	Detect and track 3D objects.
Face tracking	Detect and track human faces.
Body tracking	Detect and track a human body.
Participants	Track other devices in a shared AR session.
Anchors	Track arbitrary points in space.
point-clouds	Detect and track feature points.
Plane detection	Detect and track flat surfaces.
Meshing	Generate meshes of the environment.
Raycasts	Cast rays against tracked items.
Environment probes	Generate cubemaps of the environment.
Occlusion	Occlude AR content with physical objects and perform human segmentation.

Moreover, there exist object detection algorithms that recognize generic objects from 2D images, e.g., cars or horses, without detecting a specific instance of a predetermined image—see Redmon et al. (2016) and Hmidani and Ismaili Alaoui (2022). However, these methods are usually not capable of estimating the position and orientation of the object, which limits their applicability in 3D scenarios.

The detection of 3D objects works conceptually similarly like the detection of predefined 2D images, but is more complex due to the additional dimension. To enable the tracking of 3D objects, predefined reference objects are required. These objects can be 3D assets such as point-clouds or geometries. By continuously scanning the 3D environment, the real-world object can be detected based on the 3D object, and its position and orientation can be estimated.

In contrast to traditional image- and object recognition in 2D, there are not that many approaches for real-time 3D object recognition and pose estimation—see, for example, Chen et al. (2003) or Kazhdan et al. (2003)—and even less with generic 3D object detection and pose estimation without predefined assets (Ahmadyan et al. 2021; Simon et al. 2019). However, especially in augmented reality applications where it is important to get some context information out of the real world, it is extremely important to have real-time image- and object recognition with pose estimation, cf. (Muff and Fill 2022a).

Ahmadyan et al. (2021) showed some examples⁴ of the detection of real objects and the estimation of the object’s pose by drawing the 3D bounding box around the real objects.

4.4.1.3 Body Tracking

The concept of body tracking involves tracking humans in physical space using both 2D and 3D techniques. This can range from simply recognizing the presence of a human in the environment to identifying individual body parts and their poses, including faces and hands.

In simpler scenarios, the goal is to detect the presence of other people in an environment. In more complicated situations, additional data about the human body may be required. For example, if one wants to identify a person or gauge their emotions using 3D face recognition (Li et al. 2022).

Another very important area for body tracking is the tracking of human hands (Oberweger et al. 2015). The skeleton of human hands is very complex and there are more than 20 degrees of freedom. Hands enable precise manipulation, making reliable tracking of the entire hand a key research focus for interaction in XR applications (Schmalstieg and Höllerer 2016, p. 281). Since this is a feature that all virtual and augmented reality applications should support, most devices implement the generally defined *OpenXR* standard for hand tracking.⁵ Figure 4.15 shows an

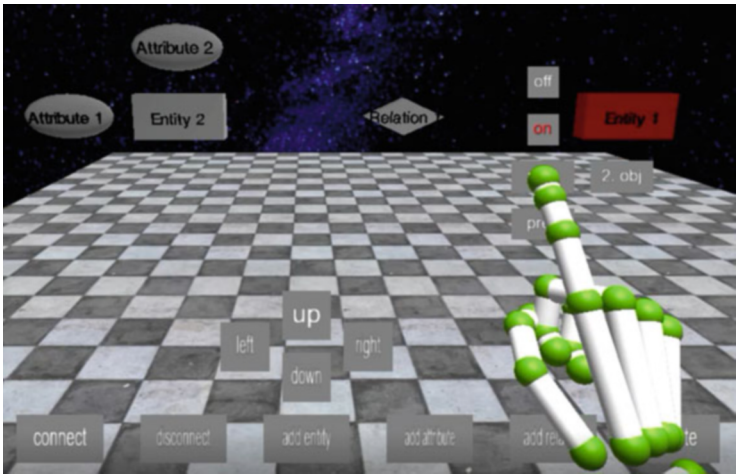


Fig. 4.15 Example of a VR application allowing the detection and tracking of human hands. Reprint from Muff (2020)

⁴ <https://github.com/google-research-datasets/Objectron>.

⁵ <https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html#XrHandTrackerEXT> last visited on: 01.03.2024.

example of a virtual reality application that uses hand tracking to interact with virtual objects. Thus, the hand tracking algorithm estimates the pose of every bone and joint in the hand.

4.4.2 *Environment Tracking*

In augmented reality applications, the tracking of the environment is equally important as the tracking of devices and objects. Certain environment tracking concepts serve as fundamental components, while others enhance the experience by providing a more realistic environment.

4.4.2.1 **Anchoring**

According to the *immersive-web* specification,⁶ an anchor is a concept that enables applications to determine the position and orientation in three-dimensional space that the underlying system will track. Anchors allow developers to establish positions in the physical world that require updates to accurately reflect the evolving understanding of the environment. This ensures that the poses remain aligned with the same positions in the physical world.

There are two concepts for anchors, which a system can use: (1) Anchors that specify a location's pose in the world, and (2) anchors that establish a relationship with semantically significant parts of the physical world that the system has detected. Thus, an anchor is a technical combination between the tracking of the **XR** device in the base coordinate system and the tracking of other poses in relation to the device and the base coordinate system.

This is not a mandatory concept for **XR** applications, but anchors can help to keep virtual objects more stable attached to the real world while a user is moving in the environment.

4.4.2.2 **Point-Cloud**

A point-cloud consists of numerous points, each possessing its own set of **3D** coordinates. By using computer vision, the sensor data of an **AR** device is examined to deduce exclusive feature points from depths, objects, edges, or patterns. These distinct feature points are then merged to produce a point-cloud map of the scanned region. To pinpoint objects in a particular location, it is imperative to locate and track unique feature points in the real environment (Kharroubi et al. 2020).

⁶ <https://immersive-web.github.io/anchors/explainer> last visited on: 01.03.2024.

This concept is not mandatory, but it can be useful for improving the stability of the environment in augmented reality applications.

4.4.2.3 Plane Detection

Plane detection is a concept to detect horizontal and vertical surfaces during the run-time of an [AR](#) experience. The feature is particularly useful for detecting physical planes on which virtual objects can be placed (Kim et al. 2017), cf. Sect. 4.4.2.5.

4.4.2.4 Meshing

Meshing is an environmental tracking technique that generates triangle meshes that correspond to the physical environment, such as walls, ceilings, or pieces of furniture. Oftentimes, these meshes are utilized as boundaries or references within the augmented reality environment, or as anchors—as previously introduced. There are different meshing techniques, e.g., based on RGB-D cameras (Lieberknecht et al. 2011).

Plane detection and meshing are not essential in [XR](#) applications, but they serve as helpful references for [AR](#) experiences. For instance, knowing the floor’s position upon initializing an augmented reality app is useful in achieving a more dynamic and realistic placement of virtual information in the real world.

4.4.2.5 Raycasting

Raycasting, also known as hit testing, involves detecting where a ray, composed of both an origin and a direction, intersects with a [3D](#) geometry.⁷ Conceptually, this process can be performed with virtual, or real-world [3D](#) geometries.

The process of real-world raycasting is intricately linked with the concepts of plane detection and meshing—see Sects. 4.4.2.3 and 4.4.2.4. Initially, it is necessary for the application to recognize the real-world objects, e.g., walls as planes. Thereafter, it becomes possible to perform raycasting against the recognized virtual [3D](#) geometries.

During this process, the calculation determines whether a ray originating from a point of origin in a [3D](#) coordinate system and traveling in a specified direction intersects with any [3D](#) object, i.e., a basic edge-face intersection test is performed (Jiménez et al. 2001). Raycasting has many possibilities for application in [3D](#) environments. Every [3D](#) desktop application utilizes raycasting to determine the correspondence between a user’s click on the [2D](#) screen and the corresponding clicked object in the [3D](#) environment. Furthermore, the use of raycasting is

⁷ <https://immersive-web.github.io/hit-test/hit-testing-explainer.html> last visited on: 01.03.2024.

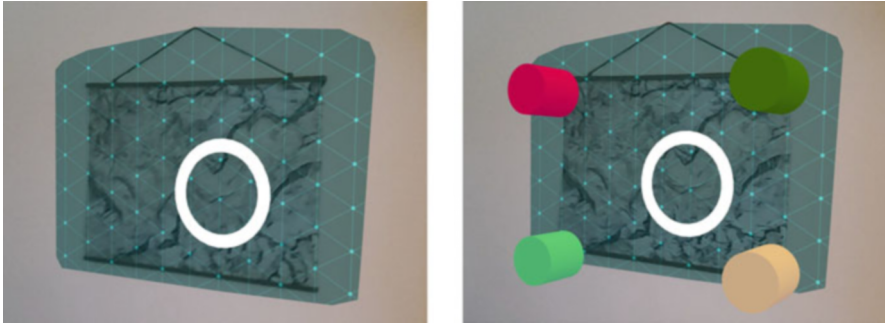


Fig. 4.16 Example of using plane detection, meshing, and raycasting to detect the surface of a drawing on a wall. The image on the left displays a plane detection example with the corresponding 3D mesh visible. The white circle indicates where an invisible ray, cast at a 90° angle from the AR device, makes contact with the detected plane. The image on the right presents additional 3D objects attached to the surface of the mesh

necessary to point at and interact with 3D objects in virtual and augmented reality applications (Jiménez et al. 2001). For example, in the hand-tracking example in Fig. 4.15, the intersection between the 3D objects of the scene with the 3D objects of the tracked hand skeleton is checked with raycasting. In addition, the example in Fig. 4.16 shows the use of plane detection, meshing, and raycasting to detect the surface of a drawing on a wall. The image on the left displays a plane detection example with the corresponding 3D mesh visible. The white circle indicates where an invisible ray, cast at a 90° angle from the AR device, makes contact with the detected plane. The image on the right presents additional 3D objects attached to the surface of the mesh.

4.4.2.6 Environment Probes

Environment probing is a technique for creating a cubemap to depict a specific region of the physical environment. Its primary purpose is to produce reflections of the environment on digital objects, providing a more realistic experience—cf. Kán and Kaufmann (2012) and Ropinski et al. (2004). Environment probing is an advanced AR feature that is not necessary for functionality and will, therefore, not be discussed further.

4.4.2.7 Occlusion

Occlusion allows virtual content to be hidden based on detected environmental or human depth, known respectively as environment occlusion and human occlusion. This augments the AR experience by concealing 3D objects that are otherwise

blocked by virtual or real elements (Kasper et al. 2017). Although highly sophisticated, this feature is not required for basic functionality.

4.4.3 Requirements for the Detection and Tracking

When considering the requirements for combining virtual and augmented reality and metamodeling, it is necessary to have concepts for the detection and tracking for the environment and real-world objects. Thus, different requirements evolve.

To determine the user's location in a **XR** environment, it is essential to consider the device's position and orientation relative to the base coordinate system.

- **GSR11:** A metamodeling environment must enable the real-time tracking of a user's position and orientation relative to the base coordinate system.

To allow for useful application scenarios in virtual and augmented reality, the detection and tracking of **2D** visual references such as images or markers must be considered.

- **GSR12:** A metamodeling environment must support concepts for the detection of **2D** images.
- **GSR13:** A metamodeling environment must support concepts for the tracking of **2D** images.

This holds not only for **2D** images, but also for **3D** objects from the real world.

- **GSR14:** A metamodeling environment must support concepts for the detection of **3D** objects.
- **GSR15:** A metamodeling environment must support concepts for the tracking of **3D** objects.

Additionally, the detection and tracking of parts of the human body is essential. This involves concepts ranging from detecting the mere presence of other humans in the environment to tracking specific limbs such as legs, arms, or hands, as well as tracking faces and eyes.

- **GSR16:** A metamodeling environment should allow for the tracking of human bodies and their specific parts.

When examining the requirements for tracking environments, the only essential feature for a **3D**-aware metamodeling environment is the raycasting concept. As mentioned by previous requirements, the inclusion of the third dimension introduces new requirements for managing **3D** objects. Raycasting can be used to aid in this management.

- **GSR17:** A metamodeling environment must allow the use of raycasting concepts.

4.5 Context

“Parts of this section have been published in a similar form as a research paper at the AAAI 2022 *Spring Symposium on Machine Learning and Knowledge Engineering for Hybrid Intelligence (AAAI-MAKE 2022)* with the title: *A Framework for Context-Dependent Augmented Reality Applications Using Machine Learning and Ontological Reasoning (Muff and Fill 2022a)*.”

As described in Chap. 1, a significant potential advantage of merging virtual and augmented reality with conceptual modeling is the ability to link conceptual knowledge back to the original real-world concept from which it originated. Considering again the model theory introduced in Sect. 2.1, models are always models of something, i.e., mappings from, representations of natural or artificial originals (Stachowiak 1973; Kühne 2006). Thus, models or individual parts of models can often be linked directly to real-world objects or locations. In traditional 2D modeling, e.g., on a desktop application, this direct link to the real world is entirely invisible.

With the incorporation of three-dimensional space and the consideration of the real-world environment as a model environment, new possibilities for metamodeling emerge and, with them, new requirements for a potential metamodeling environment that enables 3D enhanced modeling. The concepts introduced in Sects. 4.2, 4.3 and 4.4 are more technical driven and deal mainly with the main requirements that are needed for extended reality in general and in combination with metamodeling.

This section considers the more conceptual question of how the link between conceptual knowledge back to real-world objects can be achieved. Thus, in this section, we discuss various aspects and requirements relevant to this real-world context.

4.5.1 Mapping to Real-World Objects and Semantic Reasoning

Mapping of conceptual models back to real-world objects is not straightforward. As introduced in Sect. 4.4, it is possible to detect and track real objects based on certain characteristics such as color, form, or sensor data, if the exact object is known. There are also other aspects to consider. For example, when it is not an exact object that should be tracked, but rather the general context of the working situation. As a running example, imagine a scenario where a human actor performs work in a manufacturing process. To comply with workplace safety, the user should be informed about necessary safety measures, for example, wearing ear protection in loud environments (Muff and Fill 2022a).

One approach to ensuring safety is to educate employees on potential dangers in advance. This option may not always be feasible or cost-effective. Alternatively, existing documentation, ideally in the form of conceptual models, of work processes and hazard scenarios can be used to enable users to acquire this knowledge

themselves. Here again, the problem of missing knowledge about conceptual modeling may pose a problem. Thus, augmented reality can reintroduce conceptual knowledge to the practical work environment without requiring the user to possess knowledge of conceptual modeling.

A framework for creating context-aware augmented reality applications has been presented by Krings et al. (2020). The framework provides a reusable approach to facilitate the development of context-dependent AR applications for mobile phones by describing the base structures to enable context-aware adaptations of AR content. Furthermore, there are approaches for context-aware augmented reality that use machine learning or knowledge reasoning approaches (Zhou et al. 2008; Grubert et al. 2017; Hervás et al. 2011). All these approaches are very specific and do not consider general concepts for metamodeling.

For realizing context-dependent AR applications based on conceptual knowledge, we introduce a framework that contains the concepts of *machine learning*, *ontologies*, and *reasoning*. As proposed in van Harmelen and Teije (2019), there are different patterns on how to combine the concepts of machine learning and knowledge reasoning. Thereby, the two data structures *model-free data* and *model-based data* are distinguished, as well as the two algorithmic components *context reasoning* and *machine learning*. Since the information received from the sensors of AR devices is mainly in a raw format and must be further processed to get useful context information about the environment, this input is classified as *model-free data*. The model-free data can be classified into model-based data using machine learning techniques. Additionally, model-based data can be incorporated as extra input for the machine learning process to limit the classification range for the model-based output data.

After classifying the sensor data, we can utilize the information to infer further actions or suggestions for the user. This can be achieved through the use of ontologies, i.e., *knowledge reasoning*, which allows knowledge sharing, reuse, and logic-based reasoning and inference (Wang et al. 2004). Since the machine learning process generates *model-based data* as output, we can apply a second pattern described in van Harmelen and Teije (2019). This design pattern utilizes model-based data as input to the knowledge reasoning process. The output, which includes additional inferred information, is also in the form of model-based data. When combining these two patterns, the resulting design pattern is named *Learning an Intermediate Abstraction for Reasoning* (van Harmelen and Teije 2019).

The framework components and the corresponding pipeline steps are depicted in Fig. 4.17, aligning with the pattern described in van Harmelen and Teije (2019). In the context of this framework, the machine learning process corresponds mainly to the recognition of images and predefined 3D objects, as introduced in Sect. 4.4, as well as more generic recognition approaches from the computer vision area.

The framework considers the real environment and a mobile AR application. Technically, it is very difficult to get adequate information about a real environment without any knowledge about the context of the environment, e.g., some conceptual knowledge about the environment. Thus, to objectively describe the business environment within the workplace, it is necessary to provide additional information,

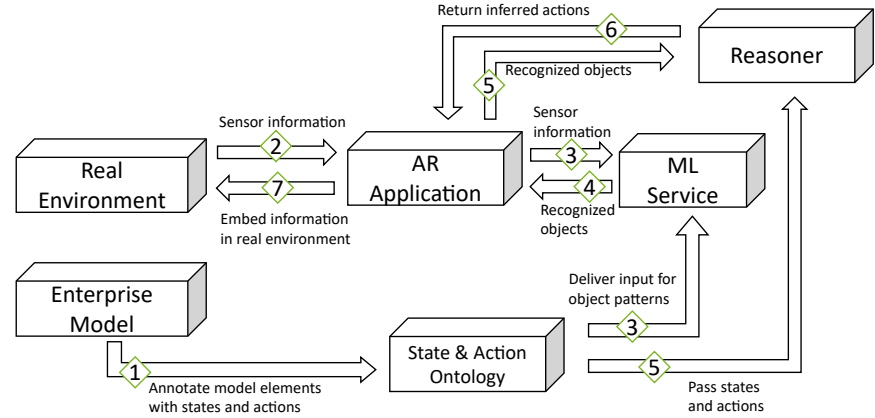


Fig. 4.17 Framework for context-dependent AR applications showing data collection and information processing using seven steps. Reprint from Muff and Fill (2022a)



Fig. 4.18 Business process model for an exemplary use case, annotated with concepts from two context ontologies. These ontologies provide additional context to the process. Two ontologies exist—one for different situations and the other for different actions. Reprint from Muff and Fill (2022a)

e.g., in the form of a business process model, annotated with additional information about the environment. For example, this could be states in the form of object patterns and actions described by an ontology (1)—see Fig. 4.18. Thereby, the information regarding the user’s context and required actions is formally presented. Subsequently, this will facilitate object recognition and action inference through a reasoner.

When the application starts, the AR device’s various sensors perceive the actual environment (2). This sensor information and the object patterns of the ontology are sent to a machine learning service (3). There, objects are recognized using the data provided by the ontology. The AR application receives the recognized objects (4). The recognized objects and the ontology states are then sent to the reasoner (5) to infer actions. The inferred actions are then transmitted to the AR application (6). Subsequently, the AR application is capable of embedding visual information in the actual environment (7) based on the inferred actions.

4.5.1.1 Technical Realization and Exemplary Use Case

To demonstrate the viability of the framework, we built a prototype using cutting-edge web technology. Specifically, we created a mobile AR environment that is independent of any particular platform. To achieve this, we utilized the JavaScript WebGL visualization framework, *THREE.js*,⁸ in conjunction with the WebXR device application programming interface (API) (Jones et al. 2023). By combining these technologies, we were able to develop applications that function across platforms with ease.

An objective of the application is to recognize objects in the real world. Currently, for privacy reasons, the *WebXR device API* lacks machine learning-based object recognition. Hence, we generated marker patterns to simulate object recognition and obtain information about the recognized objects. In the future, a machine learning service for object recognition will replace this approach. A machine learning object recognition service has the capability to identify various objects, although not all of them may be useful for our specific application. Hence, it becomes important to limit the set of identifiable objects. This can be achieved by developing an ontology that defines the states and actions relevant to the situation we aim to cover through our application. For developing and handling such ontologies, the web ontology language (OWL)⁹ and the Java-based OWL-API¹⁰ can be used.

In this prototype, markers consistently refer to ontology individuals and their corresponding type definitions. Information regarding the markers' varying visual representations is stored in a configuration file. Upon assigning types to the ontology individuals, the *Hermit* reasoner¹¹ is utilized to infer additional states and actions.

As an example scenario, consider a carpenter who produces various wood products. We will assume that the business process has been represented using the BPMN notation as shown in Fig. 4.18. Initially, we can analyze the actual real-world context of this process. For example, we may examine the *start saw* task specified in Fig. 4.18. We are aware that the use of a saw is necessary for this task and that the saw produces a loud noise. In addition, the temperature of the saw may be relevant. It is important to note that the individual operating the saw must remain stationary. A safety expert should be consulted to determine the necessary personal protective equipment and identify potential hazards and risks associated with the task.

Based on this information, we can add annotations to the process model in terms of context and safety measures (Fill 2011). For this purpose, we defined annotations with the following concepts from a specifically developed ontology: *Machine*, *PersonalProtectiveEquipment*, *PersonState*, *Sound*, *Temperature* and *Tool* as *Scene Annotations*, and *Risk*, *Hazard*, *Action* and *State* as *Action Annotations*—

⁸ <https://github.com/mrdoob/three.js>.

⁹ <https://www.w3.org/OWL/> last visited on: 01.03.2024.

¹⁰ <https://github.com/owls/owlapi> last visited on: 01.03.2024.

¹¹ <http://www.hermit-reasoner.com/> last visited on: 01.03.2024.

see the annotations in Fig. 4.18. A comparable approach to annotating workflows has been shown in Wieland et al. (2015).

Thereby, *Scene Annotations* serve as input for the AR application to determine the context of a scene. With *Action Annotations* we derive actions to be performed by the AR application based on given scenes. These annotations are formally modeled as a basis for reasoning over the concepts. For instance, in our ontology, we establish that a circular saw is a type of machine with a distinctive sawing noise that poses harm to the user due to the high level of loudness. The formal specification ontology allows us to deduce action types from situational data at a later stage. For example, if there is loud noise, the user must wear ear protection. Hence, we can infer that the appropriate action is to alert the user to wear ear protection. The classes possess various corresponding subclasses and properties to arrive at a more comprehensive ontology, enabling the differentiation of numerous distinct objects.

The relevant core ontology *classes* and *object properties* are depicted in Fig. 4.19. Each annotated concept is assumed to have a corresponding individual with an assigned class type definition and properties. It is crucial to annotate the process model and define the ontology at the onset of the development process.

Let us imagine that a user performs some tasks from Fig. 4.18, while wearing his head-mounted display (HMD) and running the AR application. As the process is not automated, the AR application is unaware of the user's current task. With the proposed framework that aims for an automated derivation of context, there is no need to explicitly define a scenario such as stating that the AR app is only functional in process step *Start Saw*. Various types of information are utilized to infer the current situation and state, and subsequently, derive appropriate actions. As a result, a dynamic assessment of the user's task can be achieved. To obtain the necessary data, the camera stream and the acceleration sensor data of the AR device are interpreted. Using machine learning-based object recognition coupled with a predefined ontology, a circular saw is detected and the situation is further analyzed—see the left side of Fig. 4.20. Markers are utilized in the initial implementation phase to represent specific objects or states. Consequently, we scan the markers to provide the requisite information for the application, functioning as a surrogate for more elaborate detection methods. Refer to the scenario depicted on the right-hand side of Fig. 4.20.

In this example scenario, two markers are used as a proxy to indicate the presence of a circular saw and the stationary position of the user to the application. Without the use of markers, this information would be derived using detection or machine learning approaches. To notify the user that the marker has been recognized, a representative image of the marker's information is displayed in the prototype. For instance, gear wheels may be depicted to represent a machine. Now the application assigns the recognized objects as individuals to a type of the ontology, e.g., the individual *Machine* to the type *CircularSaw* and the individual *PersonState* to the type *PersonStandStill*. Subsequently, this information is passed to the reasoner, which infers that the individual State is of the type *MachineUsage*, since there is a circular saw and the person is standing still.

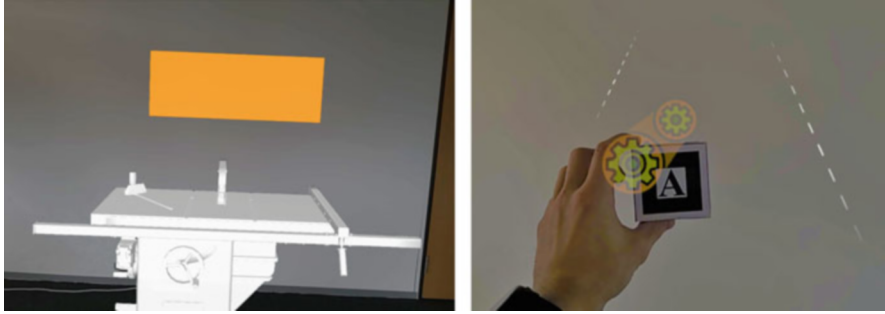


Fig. 4.20 Initial scene with a circular saw and an information pane on top of the object (left) and an example of the marker recognition (right). Reprint from Muff and Fill (2022a)

In the ontology we have the two object properties (Formula (4.1)) and (Formula (4.2)). Thus, the inferred state type can be described formally as in (Formula (4.3)):

$$\text{ObjectPropertyAssertion}(:\text{State_has_Machine} :s :m) \quad (4.1)$$

$$\text{ObjectPropertyAssertion}(:\text{State_has_PersonState} :s :ps) \quad (4.2)$$

$$\text{CircularSaw}(m) \wedge \text{PersonStandStill}(ps) \wedge \text{State}(s) \rightarrow \text{MachineUsage}(s) \quad (4.3)$$

As a circular saw is a type of sawing machinery, it can be inferred from the ontology that the specific *Sound* is classified as *SawingNoise*. This classification is defined by object properties in the ontology, as shown in Fig. 4.19. The designation of sawing noise indicates that the specific *Hazard* is associated with *Noise*, which further implies that the particular *Risk* is of a *HighRisk* type. Furthermore, the specific type of sound produced by the *SawingNoise* suggests that the *PersonalProtectiveEquipment* required is *HearingProtection*. As a result, a *WarnForHearingProtection* type of the individual *Risk* is inferred. As shown on the left side in Fig. 4.21, the warnings inferred by the application are then displayed as text in an additional object above the machine in the real world.

We can expand the use case by assigning the individual *Temperature* the type *ExtremeHighTemperature* by scanning the corresponding marker in the AR application. Again, the marker is used as a proxy instead of using more sophisticated detection approaches. The ontology will then attempt to infer more information. Therefore, the individual *Risk* is classified as *ExtremeHighRisk* and there are additional warnings to wear eye and hand protection—see the right side of Fig. 4.21.

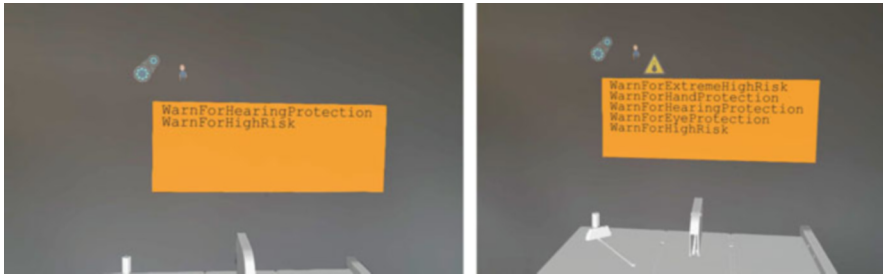


Fig. 4.21 Example of the scene when the information *PersonStandStill* and *SawingMachine* has been given to the application (left) and with the additional information *ExtremeHighTemperature* (right). Reprint from Muff and Fill (2022a)

4.5.2 Requirements for Context

Although this use case is highly simplified, it illustrates how a user can obtain support during a work process with the aid of sensor information from AR devices, through which additional context information can be inferred. This allows the information from the conceptual model to be used effectively at the appropriate time and place for an optimal AR experience. However, it is not limited to just this. Since contextual knowledge from the real environment is directly linked to the conceptual model, it is possible to infer information about the conceptual model itself, such as the current process step of an employee and the next step. This creates a bidirectional mapping between conceptual knowledge and reality. When considering the requirements for combining virtual and augmented reality and metamodeling, it is necessary to have concepts for the mapping to real-world objects and semantic inference. Thus, the following specific requirements emerge.

The mapping to real-world objects without knowing the exact appearance of the real-world object must be considered.

- **GSR18:** A metamodeling environment must consider concepts for the mapping to real-world objects without knowing the exact visual representation.

Further, it should be possible to infer contextual information about the environment.

- **GSR19:** A metamodeling environment must allow for the inference of context information.

4.6 Interaction

Human-computer interaction is a long-standing and expansive area of computer science. With the emergence of extended reality, the interaction paradigms established for traditional 2D desktops—see Dix et al. (2003)—are no longer adequate.

Thus, alternative or modified interaction techniques must be devised. Interaction can be achieved in extended reality through various means, including interaction with additional devices, such as pointing devices, interaction with gestures, or interaction with speech (Doerner et al. 2022).

4.6.1 Physical Keyboard-Based Interaction

The traditional keyboard is still one of the most important input devices in state-of-the-art computer interaction. Using a keyboard in extended reality is basically unproblematic, and the functions of the keyboard can be integrated into the virtual environment through the software interface. Nevertheless, interacting with a keyboard in a highly immersive virtual environment, such as VR, presents significant challenges. Users cannot see their hands or the keyboard itself, resulting in considerable difficulty or even impossibility in using it. Furthermore, the use of a keyboard can be highly restrictive due to its physical confinement to a desk, limiting free movement.

4.6.2 2D Mouse-Based Interaction

The traditional 2D mouse is commonly used for desktop applications and is designed for 2D environments on the screen. It functions as an interface between the screen and position in the user interface of the graphic platform. The location of the mouse pointer on the screen corresponds to a position in two-dimensional space. Integrating a 2D mouse into a 3D environment is feasible. The mouse pointer is a vector that extends the field of vision and has the possibility to collide with objects in the 3D environment (see Sect. 4.4.2.5). Interacting with 3D objects on a conventional 2D screen is relatively easy. Figure 4.22 shows an example of how interaction with 3D environments with a 2D mouse on a 2D screen looks like. With the help of a 2D mouse, the pointer on the screen can be moved. Based on that pointer position on the 2D screen, a ray is cast into the virtual 3D environment. This ray is then used to calculate intersections with objects in the 3D environment—see Sect. 4.4.2.5.

When in a virtual or augmented reality environment, mouse interaction becomes more complex, as the user has to find a way to move the mouse pointer in 3D space. Unlike a normal screen, there is no unchangeable window available in this type of environment. The use of a conventional 2D mouse in virtual and augmented reality is further complicated by the fact that the mouse is usually tied to a desk and must be seen to work with it.

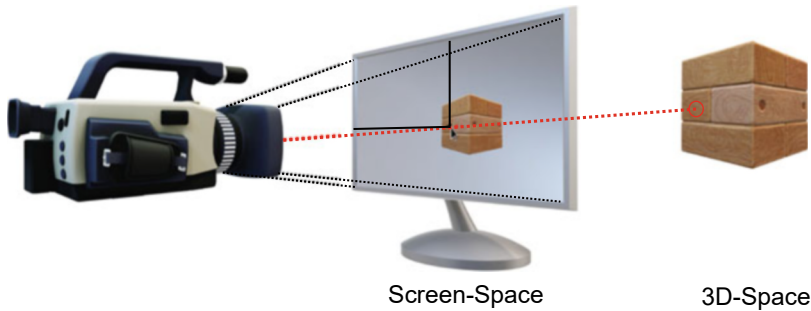


Fig. 4.22 Visual example of a 3D interaction with a 2D mouse. With the help of a 2D mouse, the pointer on the screen can be moved. Based on that pointer position on the 2D screen, a ray is cast into the virtual 3D environment. This ray is then used to calculate intersections with objects in the 3D environment

4.6.3 3D Mouse-Based Interaction

3D mice, also called *Controller*, are mostly designed for interaction in 3D space and therefore solve many of the problems of a traditional 2D mouse. For instance, 3D mice can be tracked at any time by position tracking, thus supporting six degrees of freedom. Furthermore, 3D mice can be replicated in the virtual environment with an accurate position, making physical presence unnecessary.

This opens up numerous possibilities for the use of 3D mice in highly immersive environments. 3D mice can be utilized to point to objects, and the attached buttons can be used to select objects and initiate or terminate functions. Figure 4.23 shows a visual example of the interaction with a 3D mouse controller in an augmented reality environment. The tracked position of the controllers, along with a white line that serves as the direction vector, is depicted as three-dimensional objects.

4.6.4 Voice-Based Interaction

Voice-based interaction has become a valuable option for interacting within XR environments. A suitable microphone and speech recognition software enable the creation of a powerful tool for digital device interaction. An example of such a speech assistant is the “Mozilla Deep Speech” project.¹² This allows to recognize spoken language and react to it.

One challenge that arises with speech recognition is the software’s ability to correctly interpret spoken language. The program relies on algorithms programmed into the speech recognition software to interpret speech. As a result, only the

¹² <https://github.com/mozilla/DeepSpeech> last visited on: 01.03.2024.



Fig. 4.23 Visual example of the interaction with a 3D mouse controller in an augmented reality environment. The tracked pose of the controllers are additionally visualized as 3D objects

functions incorporated by the software's developer can be successfully interpreted. The contextual interpretation of speech statements remains a difficult task.

4.6.5 *Gesture-Based Interaction*

Gesture-based interaction is a potential interaction method that is closely related to the detection and tracking capabilities of HMDs. No conventional input device is used. Instead, the interaction with the XR environment is achieved by interpreting hand gestures or, in some cases, body movements. The gesture-based interaction is often used as a substitute for 3D mouse-based interaction. Due to the intricacies of hand and finger recognition and tracking, this approach only allows for rudimentary actions such as pointing, grabbing, or pinching. The use of additional buttons, such as on 3D mice, is not possible. Figure 4.24 shows a visual example of the interaction with hands in a immersive virtual reality environment. Since the real hands are not visible, virtual replications of the tracked hands are visualized.

Another method of gesture-based interaction is eye interaction; cf. Sect. 4.4. Since this method has only limited applications, we will not elaborate further here.

4.6.6 *Collaborative Interaction*

Besides the device-based and device-less interaction methods discussed, concepts for collaborative interaction should be considered when introducing virtual and augmented reality to metamodeling. In traditional metamodeling tools, e.g.,

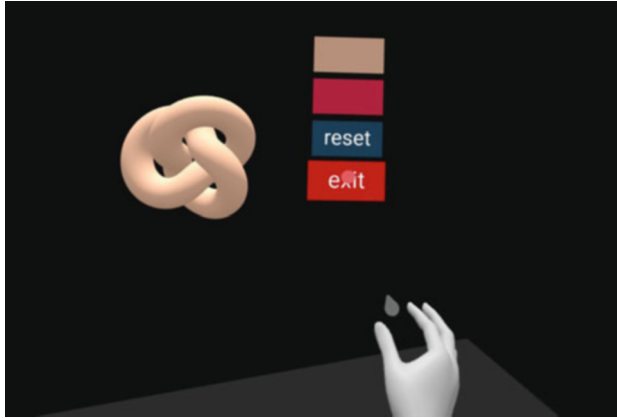


Fig. 4.24 Visual example of the interaction with hands in a immersive virtual reality environment. Since the real hands are not visible, a virtual replication of the tracked hands are visualized

ADOxx (Fill and Karagiannis 2013) or MetaEdit+ (Kelly et al. 1996), collaboration between users can be achieved. This collaboration can happen by synchronizing the models via [APIs](#) or by exporting and importing models as text files between different users. As far as we know, none of the conventional [2D](#) metamodeling platforms currently allow for collaborative modeling by multiple users on the same model. Additionally, collaboration on traditional [2D](#) displays is feasible through cooperative work in front of a shared screen to discuss and manipulate conceptual models. When considering the use cases again for collaboratively modeling a business model canvas in augmented reality (see Sect. 3), this becomes a problem. Since a user wearing a [HMD](#) cannot share what he is seeing with others, the traditional way of sharing screens between multiple users is dissolved. Thus, other concepts are needed for real-time collaboration in virtual and augmented reality.

Real-time synchronization of virtual and augmented reality environments is already possible, but there is a lack of approaches that utilize this synchronization in conceptual modeling. For example, Vogel et al. (2021) or West et al. (2010) have explored this topic to some extent, but this capability is not yet considered in metamodeling in general. Furthermore, since conceptual models may no longer be entirely separated from reality, it is essential to synchronize virtual content among various devices and adjust it to align the position of virtual objects with the real-world environment.

4.6.7 Requirements for Interaction

When considering the requirements for combining virtual and augmented reality and metamodeling, it is necessary to have concepts for interaction with or in [3D](#), [VR](#), and [AR](#) environments. Thus, different requirements evolve.

To enable seamless interaction with 3D, VR, and AR environments, appropriate devices and interaction approaches must be supported to interact with models and define metamodels.

- **GSR20:** A metamodeling environment must enable adapted interaction approaches for defining metamodels.
- **GSR21:** A metamodeling environment must enable adapted interaction approaches for interaction with models or model environments.

Since by default, different users on their own devices cannot see what others see on their devices, real-time collaboration is crucial in virtual environments.

- **GSR22:** A metamodeling environment must enable real-time synchronization of virtual information embedded into virtual or augmented reality environments.

4.7 Aggregated Generic and Specific Requirements for Joining Metamodeling with Extended Reality

Taking into account all the aspects introduced in this chapter, a comprehensive list of general and specific requirements can be presented. In the following, two tables list the nine *global generic requirements* and the 22 *global specific requirements*. The generic requirements can be found in Table 4.2, and the specific requirements can be seen in Table 4.3.

All of these requirements have at some point an impact on metamodeling, be it on a very high level, i.e., in a meta²-model, in a metamodel, or in a model. Regarding the model hierarchy introduced in Sect. 2.1.3, this means that a requirement can have an impact on the M3, the M2, or the M1-level—see Fig. 2.2. Furthermore, a modeling method (see Sect. 2.1.3.1) always consists of a *modeling technique* and *mechanisms and algorithms*. This raises the question of how the requirements introduced affect these components of a modeling method.

4.7.1 Requirements for the Meta²-Model and Modeling Methods

In this section, the implications of these requirements on specific parts of the model hierarchy and modeling methods are discussed. Thus, the five areas of *coordinate mappings*, *visualizations*, *detection and tracking*, *mapping to real-world objects and semantic reasoning*, and *interaction* are discussed in more detail.

Table 4.2 List of global generic requirements for joining metamodeling with extended reality

Generic requirement	Description
GGR1	In addition to hand gestures and voice control, other interaction options adapted for virtual and augmented reality should also be enabled.
GGR2	It must be possible to attach virtual objects to real objects by means of anchoring, i.e., reference point information must be stored somehow in the model.
GGR3	Object recognition during run-time must be enabled, e.g., by using machine learning algorithms for object recognition, as well as semantic reasoning.
GGR4	The accurate positioning of both the user and objects in the physical setting must be determined, including indoor and outdoor environments.
GGR5	To achieve an overlay of the real and the virtual objects and to annotate them with additional information, the detection of real objects must be possible.
GGR6	Based on the real environment, semantic inferences, i.e., states and consequences about the user's context must be possible. Possibilities to enable this are, e.g., ontological or rule-based reasoning.
GGR7	Real-time collaboration should be supported, whether modeling in a multi-user environment in the same location or over a distance.
GGR8	Models must support different views of the same model for different situations.
GGR9	The connection of related models and the appropriate visualization of these connections must be enabled.

4.7.1.1 3D Coordinates

Examining the specific requirements of Sect. 4.2, it is evident that these requirements are necessary for all areas of conceptual modeling in combination with extended reality. Therefore, it is essential to have concepts that allow defining traditional 2D coordinates, as well as 3D absolute and relative coordinates for all meta²-model concepts that can be visualized. Positioning and rotations adhere to the same principle. Therefore, the specific requirements GSR1–GSR5 should be considered at the meta²-level, i.e., the M3-level of the model hierarchy.

4.7.1.2 Visualizations

Taking into account the components of a modeling method introduced by Karagiannis and Kühn (2002), every modeling language has a *notation*, a *syntax*, and *semantics*. For modeling languages, the visualization of model components, e.g., in this case also the visualization of 3D objects (see Sect. 4.3), is defined in the *notation* of a modeling language. The notation can be divided into a representation part and a dynamic part, i.e., the static visualization of a model component and its dynamic

Table 4.3 List of global specific requirements for joining metamodeling with extended reality

Specific requirement	Description
GSR1	A metamodeling environment must support three-dimensional coordinates for the base coordinate system.
GSR2	A metamodeling environment must support three-dimensional relative coordinates.
GSR3	A metamodeling environment must support absolute coordinates.
GSR4	A metamodeling environment must support 3D coordinates for positioning.
GSR5	A metamodeling environment must support 3D rotations.
GSR6	A metamodeling environment must allow for the visualization of 3D objects.
GSR7	A metamodeling environment must allow the use of well-known 3D data formats.
GSR8	A metamodeling environment must allow for the definition of 3D visualization on the level of metamodels.
GSR9	A metamodeling environment must allow dynamic changes of three-dimensional visualizations.
GSR10	A metamodeling environment must support concepts to define the dynamic behavior of model components on the level of metamodels.
GSR11	A metamodeling environment must enable the real-time tracking of a user's position and orientation relative to the base coordinate system.
GSR12	A metamodeling environment must support concepts for the detection of 2D images.
GSR13	A metamodeling environment must support concepts for the tracking of 2D images.
GSR14	A metamodeling environment must support concepts for the detection of 3D objects.
GSR15	A metamodeling environment must support concepts for the tracking of 3D objects.
GSR16	A metamodeling environment should allow for the tracking of human bodies and their specific parts.
GSR17	A metamodeling environment must allow the use of raycasting concepts.
GSR18	A metamodeling environment must consider concepts for the mapping to real-world objects without knowing the exact visual representation.
GSR19	A metamodeling environment must allow for the inference of context information.
GSR20	A metamodeling environment must enable adapted interaction approaches for defining metamodels.
GSR21	A metamodeling environment must enable adapted interaction approaches for interaction with models or model environments.
GSR22	A metamodeling environment must enable real-time synchronization of virtual information embedded into virtual or augmented reality environments.

visualization changes during modeling or model execution. Since this is true for all modeling languages, these are also concepts to consider at the meta²-level, i.e., the M3-level of the model hierarchy. Therefore, the specific requirements GSR6–GSR10 must be considered on the meta²-level and concepts for static and dynamic visualization of 3D objects must be introduced.

4.7.1.3 Detection and Tracking

When examining the requirements outlined in Sect. 4.4, it is not immediately clear how they fit into the larger picture. Technologically speaking, there must be methods to detect and track various entities in order to enable these features in metamodeling. If these technological capabilities are in place, the question arises as to where these concepts should fit within the model hierarchy. There are numerous possibilities to consider.

First, if these requirements apply to all modeling techniques in general, i.e., either modeling languages or modeling procedures, then the concepts of detection and tracking must be introduced at the meta²-level. Second, when such concepts are used more for, or in modeling languages or modeling procedures, i.e., as mechanisms or algorithms, it is not immediately clear where to consider these features. Refer to Fig. 2.4 for an illustration of how mechanisms and algorithms can be classified within three categories: *Generic*, *Specific*, and *Hybrid* algorithms. Generic mechanisms are implemented on top of the meta²-model to enable usage in all metamodels based on the meta²-model. Specific mechanisms are designed for particular metamodels. Hybrid mechanisms are created on the meta²-model, but are tailored to specific metamodels to enhance usability, as noted by Karagiannis and Kühn (2002) in their work on metamodeling platforms.

GSR11 and GSR17 are considered fundamental aspects of any 3D, VR or AR application and will not be discussed further. However, the detection and tracking of 2D images, 3D objects, and body parts, i.e., GSR12–GSR16, needs to be looked at more closely. When investigating use cases in which detection and tracking are essential, it is evident that the process mainly involves detecting and tracking instances. For example, when examining the IT infrastructure scenario in Fig. 3.9, the real-world servers being detected and tracked are always specific instances, not a concept of “server” in general, i.e., a concept in a metamodel. This also applies to the business perspective case in Fig. 3.7, where the system must recognize and trace the location of the subsequent individual step.

Thus, the requirements for detection and tracking, i.e., GSR12–GSR16, are not to be considered in the meta²-model, but rather in metamodels with specific language concepts that can then be processed by mechanisms and algorithms. Specifically, the author suggests incorporating these concepts into metamodels as needed, i.e., at the M2-level of the model hierarchy.

A proposal on the utilization of these concepts at the metamodel level will be presented in Chap. 5. Ideally, these language concepts would not be defined by the metamodeler creating a new modeling language, but globally at the level of a

metamodeling platform. This means that these language concepts are not defined at the meta²-level, but are predefined language concepts on the meta-level that can be reused by other metamodelers for their metamodels as needed. This is sort of a hybrid approach. The language concepts would be available at any time and globally defined, but would not be considered in the meta²-model and would be processed by hybrid mechanisms and algorithms at run-time.

4.7.1.4 Context

This section discusses the particular needs for context, i.e., mapping to real-world objects and semantic reasoning, specifically GSR18 and GSR19. Like the requirements for detection and tracking, the requirements for mapping and reasoning involve instances of models. At the meta²-level, the context of a situation cannot be generally defined, since it depends on sensor information at run-time. Mechanisms and algorithms process environmental and model data throughout modeling and model execution. It is possible that these mechanisms and algorithms exist as generic algorithms in the meta²-model—see Sect. 6.1.1.8. It is also possible that mechanisms or algorithms, e.g., for the inference of the current task in a BPMN process, are implemented in a completely independent application that only takes models as input data and is thus completely decoupled from the modeling method. Thus, it is recommended that GSR18 and GSR19 should be linked to a modeling language or execution environment.

4.7.1.5 Interaction

Interaction often depends heavily on the available device technology and software. For instance, determining whether a virtual 3D object can be manipulated using a controller or bare hands during a virtual modeling process (GSR21) or how to interact with the 3D visualization of a metamodel component's notation while defining a metamodel (GSR20) is primarily dependent on the software used to perform these tasks. It is not dependent on the modeling method or the meta²-model, i.e., a metamodeling platform or a model execution platform.

This is not entirely accurate for GSR22, which refers to real-time collaborations. While modeling platforms commonly include user structures to manage user rights, the M3-level should consider incorporating certain concepts of user rights since it is a universal concept applicable across all modeling methods, i.e., at the meta²-level. When it comes to real-time collaboration, the question arises on which level this collaboration takes place.

Collaboration can occur during the specification of the metamodel, such as for defining notation or syntax. Since the definition of syntax is independent of the newly introduced 3D concepts, it is not considered here. However, the

notation definition (refer to Sect. 4.3) must be taken into account. For instance, two users can cooperate in designing a meta-concept's 3D visualization in a 3D environment.

Additionally, collaboration can occur at the model level, such as in collaborative model creation as introduced in the collaborative business model canvas (BMC) modeling use case (see Chap. 3). It would be reasonable to have a universal user management concept that is directly linked to the meta²-model. The implementation of real-time synchronization between different users is a matter related to the underlying modeling platform, rather than the meta²-model or the modeling method.

The final option involves real-time collaboration during model execution. Consider revisiting the scenario of a business process execution with the assistance of an AR application that visually guides the user to the next task location, as described in Chap. 3. If two users are required to execute the task and both require access to the same 3D visualizations at the same time in the same place, it must be synchronized between them in real-time. In this case, the synchronization would be separate from any metamodeling platforms, including the meta²-model. It is feasible to combine these execution engines with an underlying metamodeling environment. However, such a platform does not yet exist.

4.7.2 *Need for Knowledge-Based Virtual and Augmented Reality Approaches*

After evaluating the implications of all the specific requirements presented above regarding potential changes in metamodeling, it is evident that many of these specific requirements are not applicable at the M3 level, i.e. in the meta²-model. Instead, they should be considered on the level of modeling methods. This brings us back to the particular area of *knowledge-based VR/AR*, as discussed in Sect. 2.3.6.1—see Fig. 2.20.

Within this category, one potential area of interest lies within the field of developing augmented reality applications, particularly in the context of model-based augmented reality applications. This includes design-time aspects, i.e., modeling of VR/AR applications, as well as run-time aspects, i.e., running VR/AR applications based on models. Thus, the upcoming chapter will introduce a domain-specific visual modeling method that enables the creation of augmented reality scenarios in a generic manner, incorporating many of the recommended requirements to be coupled to a modeling method.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



Chapter 5

ARWFMM: A Modeling Method as an Example for Knowledge-Based Virtual and Augmented Reality



Parts of this chapter have been published in a similar form as a research paper in: *Conceptual Modeling. ER 2023. Lecture Notes in Computer Science, vol 14320* with the title: *A Domain-Specific Visual Modeling Language for Augmented Reality Applications Using WebXR* (Muff and Fill 2023c).

Augmented reality is based on the three core concepts (1) *Detectables/Trackables*, (2) *Coordinate Mappings*, and (3) *Augmentations*—cf. Sect. 2.2.2.1. In addition, AR applications consider workflow concepts that enable dynamic changes in the AR environment based on triggers, conditions and actions.

Creating such dynamic augmented reality applications requires today advanced programming skills, e.g., for platforms and APIs such as Vuforia,¹ ARKit,² Google ARCore,³ or MRTK.⁴ To facilitate the creation of AR applications, several proposals have been made in model-driven engineering and conceptual modeling. Among these are XML and JSON schemas that describe AR scenes in generic, platform-independent formats (Ruminski and Walczak 2014; Lechner 2013) or with an emphasis on learning experiences (Wild et al. 2014). Domain-specific languages for creating AR model editors using Vuforia, ARKit, or MRTK (Ruiz-Rube et al. 2020; Campos-López et al. 2021; Seiger et al. 2021); or a BPMN extension for representing process information in AR using the Unity platform (Grambow et al. 2021).

In addition, commercial low-code and no-code tools are available to empower non-technical users with the capability to create AR applications. Examples of such

¹ <https://library.vuforia.com/> last visited on: 01.03.2024.

² <https://developer.apple.com/documentation/arkit/> last visited on: 01.03.2024.

³ <https://developers.google.com/ar> last visited on: 01.03.2024.

⁴ <https://github.com/Microsoft/MixedRealityToolkit-Unity> last visited on: 01.03.2024.

tools are UniteAR⁵ and Adobe Aero.⁶ However, these tools are mostly designed for creating a single AR scene or very simple workflows.

What is currently missing is a visual modeling approach that can represent complex AR workflows for different application scenarios, that can be easily adapted to new requirements, and that is based on open standards. To enable the creation of augmented reality applications that take advantage of the accessibility, portability, interoperability, and openness of the web, this chapter introduces a domain-specific modeling method based on models that conform to the W3C WebXR device API recommendation, allowing the definition of different scenarios such as assembly processes, maintenance tasks, or learning experiences. It should be noted that the approach presented in this chapter is firmly grounded in the field of *knowledge-based VR/AR*—see Sect. 2.3.6.1 and Fig. 2.20. This encompasses modeling VR or AR applications as well as the execution of VR or AR applications based on models and, in particular, the discovered area of *Concepts and Languages* in Sect. 2.3.

The language development of the method adheres to the guidelines for domain-specific modeling language (DSML) development by Frank (2013). The ADOxx metamodeling platform (Fill and Karagiannis 2013) was used to implement the DSML and the new modeling method was applied to a furniture assembly use case. In the initial evaluation, a feature comparison with similar languages in the field of augmented reality is performed (Siau and Rossi 2011), and a formal evaluation with the FDMM formalism is conducted (Fill et al. 2012a,b, 2013).

This chapter is organized as follows. In Sect. 5.1, we analyze previous related approaches in the area of *knowledge-based VR/AR*, and more specifically in model-driven engineering and conceptual modeling in the context of AR. Section 5.2 introduces foundational knowledge of the most important development platforms for augmented reality applications, as well as different metamodeling platforms. This information will enable the deduction of generic and specific requirements for a modeling method for *knowledge-based VR/AR*. Furthermore, knowledge of related approaches is needed to understand the subsequent decisions during the development process of the new modeling method. From the insights of Sects. 5.1 and 5.2, we derive generic and specific requirements for a domain-specific visual modeling language for AR applications and present its specification and implementation in Sect. 5.3. This is followed by a first evaluation of the modeling method in Sect. 5.4, including a use case (Sect. 5.4.1), a feature comparison (Sect. 5.4.2), and a formal evaluation (Sect. 5.4.3). Finally, in Sect. 5.5, we look at newly gained insights and point out the drawbacks of this approach for state-of-the-art 2D modeling tools.

⁵ <https://www.unitear.com/> last visited on: 01.03.2024.

⁶ <https://adobe.com/products/aero.html> last visited on: 01.03.2024.

5.1 Related Approaches for Conceptual Modeling and Model-Driven Engineering for AR

Several approaches have explored the application of conceptual modeling and model-driven engineering for augmented reality applications. In a comprehensive literature analysis, we previously identified 201 relevant papers at the intersection of conceptual modeling and VR and AR and derived the main research streams in these areas (see Sect. 2.3). From the results of this study, we have identified crucial contributions in the field of *Concepts and Languages*, which includes the area of model-driven engineering and conceptual modeling for AR. The most important contributions in this area will be concisely characterized in the following.

Ruminski and Walczak (2014) described a new text-based approach called *CARL*, which is a declarative language for modeling dynamic, contextual augmented reality environments. They state that *CARL* can simplify the process of creating AR experiences by allowing developers to create reusable, modular components which can be combined to form more complex scenes. The approach involves selecting and merging content and rules from different service providers for creating AR scenes without the need to switch between services. They demonstrate the effectiveness of the language through the implementation of a prototype AR application of a bookstore AR service. Their development approach is based on textual modeling and does not include a visual representation.

Wild et al. (2014) focused on data exchange formats for AR experiences in manufacturing workplaces. They propose two textual modeling languages that include the definition of learning activities (activityML) and the definition of workplaces (workplaceML). In addition, they discuss the challenges of implementing such a framework, including the need for interoperability between different systems. The article highlights the importance of data exchange formats for AR learning experiences in manufacturing workplaces. Based on this work, a new IEEE standard for *Augmented Reality Learning Experience Models* has been developed (Wild et al. 2020), which includes a reference implementation.⁷ It enables the direct definition of learning workflows within an AR context. The textual models for these workflows are stored only at run-time, precluding a definition outside the tool. Screenshot of this reference implementation for modeling AR workflows are available in Wild et al. (2020).

A similar approach has been developed by Lechner (2013). He proposes the XML-based *Augmented Reality Markup Language* (ARML 2.0) to describe virtual objects, their appearance, and anchors in an AR scene in relation to the real world. The paper highlights the need for standardization in the AR industry and the potential benefits of ARML 2.0 as a common data format for AR applications. Further, the separation of concerns was discussed. Lechner compared other AR browser formats to ARML 2.0 and discussed the gaps for standardizing ARML

⁷ <https://github.com/WEKIT-ECS/MIRAGE-XR> last visited on: 01.03.2024.

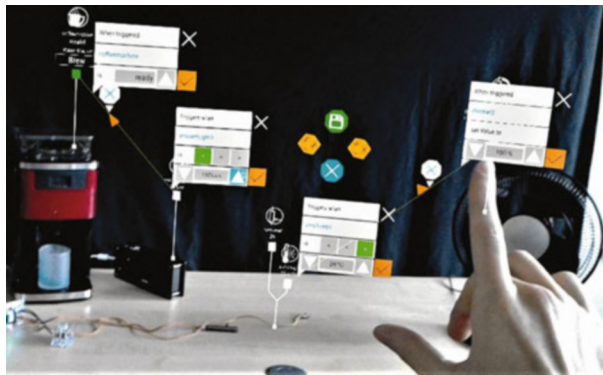


Fig. 5.1 Screenshot of the HoloFlows application for modeling IoT processes in augmented reality. Reprint from Seiger et al. (2021)

2.0. ARML 2.0 has been included in a standard issued by the *Open Geospatial Consortium*⁸ in the form of an XML grammar.

Ruiz-Rube et al. (2020) argue that bodily-kinesthetic abilities are not currently supported in common modeling tools. Thus, they proposed a model-driven development approach to create AR-based model editors, aiming at more efficient means of creating and editing conceptual models in AR. Thus, the generated applications target modeling itself. They demonstrate their approach by a tool called *ARE4DSL*,⁹ claiming that this approach has the potential to support teaching and work with models in an innovative way. It only allows for the definition of AR-based modeling applications and not for the definition of other types of AR applications.

Seiger et al. (2021) presented *HoloFlows*, a modeling approach for creating Internet of Things (IoT) processes in augmented reality environments—see example in Fig. 5.1. They argue that HoloFlows can help to address the challenges of IoT process modeling by providing a more intuitive and immersive way to design and analyze IoT systems. The approach includes the use of a mixed reality interface that allows users to visualize and manipulate IoT devices and their interactions in real-time without the need of process modeling knowledge. The approach is specific to the IoT domain and modeling is only possible within the provided AR application on the Microsoft HoloLens.

Grambow et al. (2021) introduced an approach called *BPMN-CARX*. It stands for a solution integrating context-awareness, visual AR support, and process modeling in BPMN of Industrial Internet of Things (IIoT) processes. The approach allows one to extend business process management software with AR and IIoT capabilities. Furthermore, it supports the modeling of context-aware and AR-enabled business processes. BPMN-CARX extends BPMN with new elements, including a graphical

⁸ <http://docs.openegeospatial.org/is/12-132r4/12-132r4.html> last visited on: 01.03.2024.

⁹ <https://github.com/spi-fm/ARE4DSL> last visited on: 01.03.2024.

notation, and conforms to current business process management norms without requiring a completely new notation. The approach supports the integration of AR and IoT context information during process modeling and the use of real-time sensor data for rule execution. Additionally, the framework supports AR integration during business process enactment without the need for AR users to switch between software solutions or platforms. A case study in a simulated smart factory environment demonstrated the feasibility of the approach. It is specific to business process modeling and does not seem applicable to other scenarios.

Campos-López et al. (2021) and Brunschwig et al. (2021) proposed an automated approach for constructing AR-based interfaces for information systems using model-driven and software language engineering principles without the need for coding knowledge. They introduced a model-driven approach for AR interface construction, where the interface is automatically generated from a high-level domain metamodel of the system and includes AR features like augmentations, a mechanism for anchors based on real-world position, or the recognition of barcodes and quick response (QR) codes. Additionally, it is possible to define API calls to be performed upon certain user interactions, e.g., the creation of objects. The approach is mainly designed for modeling systems that use AR, but there is no possibility to define states or executable workflows. They demonstrate the feasibility of their approach through a prototypical iOS app called *Alter*¹⁰ that is based on Apple's ARKit. Screenshots of an example of the modeling process with this prototypical application are available in Brunschwig et al. (2021).

In summary, there are existing approaches for (1) creating particular AR applications founded on models and schemata, (2) generating AR-based modeling tools using model-driven engineering, and (3) modeling AR applications that are based on conceptual modeling languages, cf. aspects of pairing conceptual modeling with VR/AR in Sect. 2.3.6.1. To our knowledge, currently there is no visual modeling approach available to represent executable AR workflows for varied application scenarios that is based on open AR standards. The next section introduces development platforms for augmented reality applications, as well as different metamodeling platforms. This is needed to define then the requirements of a modeling method and its implementation, along with an exemplary use case in the following sections.

5.2 Related Development and Metamodeling Platforms

Various methods and development platforms have been used in both research and industry to create augmented reality applications. This section outlines some of the most commonly used development platforms, including their advantages and disadvantages. Additionally, this section introduces the most used metamodeling

¹⁰ <https://alter-ar.github.io/> last visited on: 01.03.2024.

platforms, since the methodology for defining domain-specific modeling languages by Frank (2013) defined the use of a metamodeling language as a generic requirement for DSML development.

5.2.1 Development Platforms

There are numerous development platforms available for creating virtual and augmented reality applications. A majority of these platforms are proprietary and impose charges for commercial use.

5.2.1.1 Unity

The Unity development platform¹¹ is a leading development platform used to create applications for virtual and augmented reality. Unity offers a complete framework that enables developers to integrate VR and AR functionality seamlessly. The platform facilitates widespread application deployment by supporting various types of VR and AR devices, e.g., Microsoft HoloLens, Meta Quest, as well as smartphones and tablets, using the AR Foundation package.¹² Unity's scripting API and user interface streamline the development process for beginners and advanced developers.

Unity, despite its widespread use, encounters significant obstacles. Achieving good performance, particularly in resource-intensive virtual and augmented reality applications, requires a great effort from developers. Furthermore, financial constraints for independent and smaller teams are caused by the licensing structure, since advanced features require subscription fees.¹³ Additionally, the steep learning curve of the platform can create challenges for new and inexperienced users. Lastly, interoperability should also be taken into account, since Unity's scripting relies on the C# programming language, potentially limiting collaboration with other languages or platforms.

5.2.1.2 Unreal Engine

Unreal Engine, developed by Epic Games, is a widely-used game development toolset for creating 3D experiences on platforms such as PC, console, mobile, virtual

¹¹ <https://unity.com/> last visited on: 01.03.2024.

¹² <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.0/manual/index.html> last visited on: 01.03.2024.

¹³ <https://unity.com/pricing> last visited on: 01.03.2024.

and augmented reality.¹⁴ The engine offers comprehensive support for OpenXR¹⁵ and all hardware vendor APIs, from HoloLens to ARCore to Oculus.

Like Unity, the Unreal Engine presents a challenging learning curve and is not royalty-free, requiring developers to pay fees based on generated revenue.

5.2.1.3 Vuforia

Vuforia¹⁶ is a comprehensive SDK designed to empower developers in creating robust AR applications across multiple platforms. By leveraging advanced computer vision techniques, such as image recognition and tracking, the tool seamlessly integrates digital content into the real-world environment. Vuforia provides a range of target types, including images, objects, and environments, that facilitate various use cases for applications (see tracking and detection in Sect. 4.4). Vuforia has cross-platform compatibility with both iOS and Android platforms, which amplifies its accessibility for developers. Furthermore, Vuforia provides different development environments, but is mostly used on the basis of the Unity development platform described above.

Therefore, Vuforia shares the drawbacks outlined in Sect. 5.2.1.1. Additionally, Vuforia's primary focus is on mobile devices, and its tracking capabilities on HMDs are limited. Like Unity and Unreal Engine, Vuforia is a closed-source platform with a challenging learning curve and a complex pricing structure.¹⁷

5.2.1.4 ARKit

ARKit is an Apple-developed augmented reality framework that enables developers to design AR applications for iOS devices such as iPhones and iPads, as well as the Apple Vision Pro HMD. ARKit streamlines AR experience building by incorporating device motion tracking, world tracking, scene understanding, and display features. Apple built a whole development ecosystem by providing Xcode¹⁸ and the Reality Composer Pro.¹⁹ Furthermore, it is also possible to develop applications in Unity or Unreal Engine using the ARKit library on top.

There are several challenges and drawbacks associated with using ARKit. Creating an AR app with ARKit can be challenging, especially for novice developers due to its complexity. Additionally, ARKit is exclusive to iOS devices, reducing the range of users who can access AR apps developed using ARKit. However, ARKit

¹⁴ <https://www.unrealengine.com/> last visited on: 01.03.2024.

¹⁵ <https://www.khronos.org/openxr/> last visited on: 01.03.2024.

¹⁶ <https://library.vuforia.com/> last visited on: 01.03.2024.

¹⁷ <https://www.ptc.com/en/products/vuforia/vuforia-engine/pricing> last visited on: 01.03.2024.

¹⁸ <https://developer.apple.com/xcode/> last visited on: 01.03.2024.

¹⁹ <https://developer.apple.com/augmented-reality/tools/> last visited on: 01.03.2024.

remains a popular choice for developers due to its compatibility with millions of iOS devices and an easy-to-use interface for Apple devices.

5.2.1.5 ARCore

ARCore is a proprietary Google-developed [SDK](#) that allows developers to create augmented reality applications for Android, iOS, Unity, and the web.²⁰ One of the main advantages of ARCore is that it is available on a wide range of Android devices, e.g., Android-based smartphones, tablets, and [HMDs](#), making it accessible to a large audience. ARCore provides different [APIs](#) for multiple development environments such as Unity, XCode, Unreal Engine, or Android Studio. This enables a variety of target devices. ARCore, as well as ARKit, implement the guidelines proposed by the [AR Foundation](#) (see Sect. 4.4).

Even though ARCore allows the development of augmented reality applications for multiple device platforms, there is still the use of different development environments for the different devices. Thus, ARCore is not really platform-independent. Further, the development is similarly complex as on the other introduced platforms, and in comparison to ARKit does not allow for object tracking yet.

5.2.1.6 WebXR

WebXR is a web-based open-source [API](#) that enables developers to generate immersive experiences for both augmented reality and virtual reality devices (Jones et al. 2023). One of the main advantages of WebXR is that it utilizes HTML and JavaScript, facilitating the development of [VR](#) and [AR](#) apps by web developers without necessitating knowledge of new programming languages.

WebXR is only an [API](#) and is thus always based on a base [3D](#) library such as [THREE.js](#)²¹ or [Babylon.js](#).²² Furthermore, WebXR apps can run on a wide range of [XR](#) devices, such as [HMDs](#), smartphones, or tablets, which enhances its availability to users, making the [API](#) platform-independent. In theory, any mobile device equipped with a web browser and a camera can access WebXR applications without requiring device-specific adaptations of program code or specific deployment.

There are some drawbacks to utilizing WebXR. For example, WebXR is a relatively new technology, resulting in fewer resources for developers compared to established platforms like Unity. Moreover, WebXR applications may have a lower performance than native applications, which may restrict the potential types of experience that can be developed. Additionally, it should be noted that WebXR currently supports fewer [AR Foundation](#) features compared to other approaches.

²⁰ <https://developers.google.com/ar> last visited on: 01.03.2024.

²¹ <https://github.com/mrdoob/three.js> last visited on: 01.03.2024.

²² <https://github.com/BabylonJS/Babylon.js> last visited on: 01.03.2024.

Table 5.1 Comparison of the supported *AR Foundation* 6.0 features by ARKit, ARCore, and WebXR. The addition (exp.) means that the feature is available but only in experimental mode. Apple ARKit supports all the features outlined by the *AR Foundation*. Google ARCore supports all features except for four, lacking the capability to detect predefined 3D objects. WebXR implements all but five features and also lacks the crucial function of detecting predefined objects, but allows for image detection

Feature	ARCore	ARKit	WebXR
Session	✓	✓	✓
Device tracking	✓	✓	✓
Camera	✓	✓	✓
Plane detection	✓	✓	✓ (exp.)
Image tracking	✓	✓	✓ (exp.)
Object tracking	✗	✓	✗
Face tracking	✓	✓	✗
Body tracking	✗	✓	✗
Point clouds	✓	✓	✗
Raycasts	✓	✓	✓
Anchors	✓	✓	✓
Meshing	✗	✓	✓ (exp.)
Environment probes	✓	✓	✓ (exp.)
Occlusion	✓	✓	✓ (exp.)
Participants	✗	✓	✗

Nonetheless, WebXR holds significant potential as a tool for creating immersive web experiences. Its open-source nature and accessibility make WebXR particularly interesting for research and industry, and it has great potential to be adopted as the industry standard for the development of virtual and augmented reality applications.

5.2.1.7 AR Foundation Feature Comparison

Table 5.1 shows a comparison of the *AR Foundation* features supported by ARKit, ARCore, and WebXR. As is evident, Apple ARKit supports all features outlined by the *AR Foundation* in version 6.²³ Google ARCore supports all features except for four, lacking the capability to detect predefined 3D objects. WebXR implements all but five features and also lacks the crucial function of detecting predefined objects, but does allow for image detection. This feature is currently in the experimental phase and is limited to tablets and is not available for HMDs. What all three SDKs have in common is the need for programming knowledge to develop VR or AR applications. Additionally, due to privacy concerns, accessing the camera stream of an AR application is currently not possible on most standard HMDs. Real-time image and object detection are essentially excluded from HMDs until clear legal regulations on privacy are established.

Although WebXR has a limited feature set compared to ARKit and ARCore, it has been chosen as the main technology for the majority of AR and VR implementations in this work due to its open-source and platform-independent

²³ <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.0/manual/index.html> last visited on: 01.03.2024.

nature. If the mentioned drawbacks of WebXR are resolved in the future, this technology will clearly have an advantage over other closed-source approaches.

5.2.1.8 Low-Code and No-Code Platforms

In addition to the aforementioned technical **SDKs**, there exist commercial low-code and no-code platforms for generating **VR** and **AR** experiences. For instance, platforms like UniteAR²⁴ and Adobe Aero²⁵ provide a high-level way to define **VR** or **AR** scenes, albeit with limited options for incorporating more intricate logic into the experience. Thus, these methods are better suited for straightforward scenarios, but not for intricate use cases as presented in Chap. 3.

5.2.2 Metamodeling Platforms

A metamodeling platform is a software program that enables a modeler to create a modeling method, or metamodel, on a platform-specific meta²-model—see Sect. 2.1.3. The meta²-model of a platform provides the specific components for a metamodel on that platform. There are various metamodeling platforms that are based on distinct architectures and meta²-models. According to Karagiannis and Kühn (2002), a metamodeling platform must have a distributable, scalable, and component-based architecture to provide value—see Fig. 5.2. Thus, a complete metamodeling platform must always incorporate the following components.

Data storage is always achieved by a *persistence service* providing technologies such as databases and file storage to store model and metamodel data.

The meta²-model furnishes the fundamental ideas for building metamodels, and mechanisms and algorithms. Common concepts are relations, classes, attributes, and others. The meta²-model is the core of each metamodeling platform architecture, as it provides the conceptual basis and is linked to all other components.

The **metamodel base** holds all the data regarding the metamodels that the modeling platform is currently managing. Any changes to the metamodel base are passed on to the model base to ensure that the models and their associated metamodels remain synchronized.

The **mechanism base** contains data regarding the functionalities that can be applied to models and metamodels. These functionalities can be stored either in the mechanism base itself or outside of the metamodeling platform.

The **model base** contains all models which are based on the metamodels. It interacts with the metamodel base to monitor any alterations to the metamodels and to pass them on to the relevant models.

²⁴ <https://www.unitear.com/> last visited on: 01.03.2024.

²⁵ <https://adobe.com/products/aero.html> last visited on: 01.03.2024.

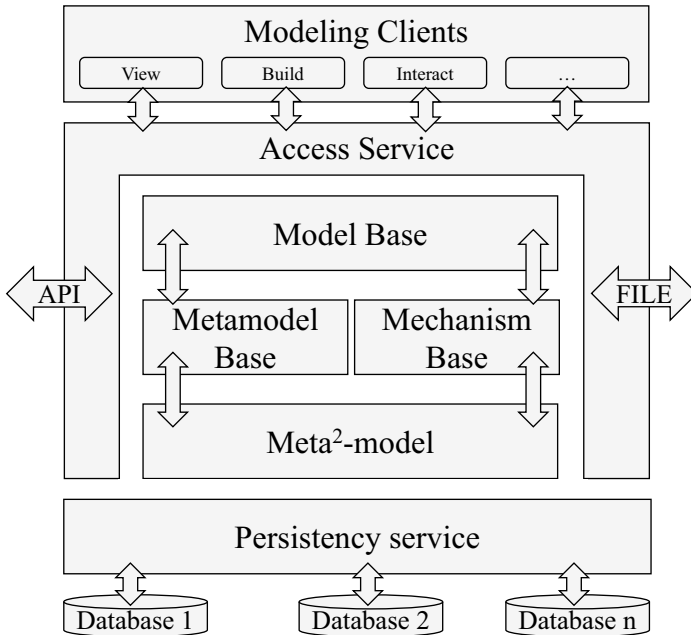


Fig. 5.2 Generic architecture of metamodeling platforms. Adapted from Karagiannis and Kühn (2002)

Access services provide interfaces to the different other bases. They manage information to steer access to the appropriate information from the bases, e.g., if they can be queried, changed or deleted by a user.

Modeling clients are on top of access services. They can be services or applications to view, build, or interact in any other form with models, metamodels, or algorithms and mechanisms.

In research and industry, there are various metamodeling platforms, each with its own strengths and weaknesses. In the following, we introduce the most important metamodeling platforms in regard to this work. It is important to note that the list of platforms provided is not exhaustive, and there are many more platforms that will not be discussed in this context.

5.2.2.1 ADOxx

ADOxx is a metamodeling platform developed at the University of Vienna in 1995 and was later transferred to the BOC Group (Fill et al. 2012a). ADOxx originated from the creation of a business process modeling toolkit known as ADONIS (Junginger et al. 2000).

ADOxx allows the creation of modeling methods, as well as the definition of model instances (Fill and Karagiannis 2013). The ADOxx platform is composed of two primary components: (1) The development toolkit and (2) the modeling toolkit. The development toolkit enables a metamodeler to construct a modeling method by defining its modeling technique, as well as mechanism and algorithms—see Sect. 2.1.3.1. The modeling toolkit then allows the modeler to employ the metamodels and create models that conform to these metamodels. The meta²-model of ADOxx is implemented in the programming language C++.

The ADOxx metamodel is an instance of the meta²-model that was created by the ADOxx developers. The ADOxx Library Language (ALL) was used to construct the metamodel. Metamodelers can use the ADOxx metamodel to create their own domain-specific metamodel via the ADOxx Definition Language (ADL). The models created in the ADOxx modeling toolkit are instances of the metamodel developed by the metamodeler. ADOxx’s meta²-model, depicted in Fig. 5.3, provides the structure for the creation of metamodels.

The core of this meta²-model consists of *classes* and *relationclasses* that are organized by *model types*. The cardinalities indicate that each *model type* must have at least one *class* assigned to it. Furthermore, *model types* can be further specified by *views*, which limit the *classes* and *relationclasses* that are displayed. Both *classes* and *relationclasses* have *attributes* that can be specified by *facets* such as a help text or regular expressions to further restrict the values of the *attributes*. Two kinds of class attributes are specified. (1) *Notebook definitions* specify the attribute representations and are written in the *ATTRREP* grammar. *ATTRREP* defines which attributes are visible in the dialogues of modeling objects

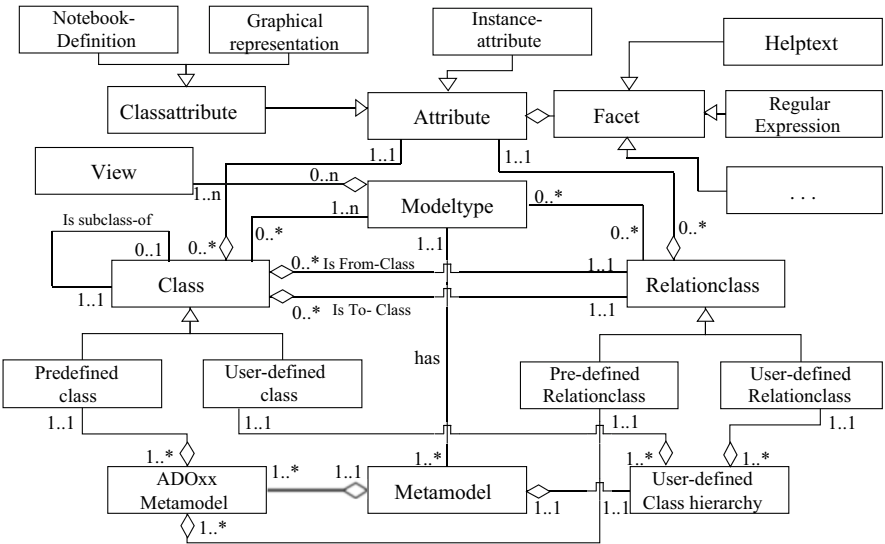


Fig. 5.3 The ADOxx meta²-model. Adapted from Fill and Karagiannis (2013)

to the modeler. (2) The graphical representations that determine the dynamic visual representation of *classes* and *relationclasses* in the GRAPHREP grammar. There are different types of instance attributes, such as single or multiple value data types including string, integer, float, as well as special data types *INTERREF* and *record class*. *INTERREF* is an important concept to connect objects instances in the same or other model instances, or whole model instances. Tables can be created via the *record class* concept, which are collections of *attributes* with the data types mentioned above. Classes support a hierarchical concept, which allows one to create subclasses. *Relationclasses* are meant to connect two classes by a relationship. Each *relationclass* must have exactly one *from-class* and one *to-class* (Fill and Karagiannis 2013).

The concepts described above are the most important concepts of the ADOxx meta²-model. There are many more concepts, which we will not describe in this work. Based on the ADOxx meta²-model, the ADOxx platform provides many functionalities to the user which allow one to create modeling methods. The architecture of the ADOxx metamodeling platform is in accordance with the generic architecture of metamodeling platforms introduced in Fig. 5.2. Thus, ADOxx provides different components for handling and visualizing models on the basis of the supplied metamodels, e.g., visual modeling or table-based modeling. Furthermore, it is possible to analyze, simulate, and evaluate models, as well as to transform models. Finally, it is possible to define different export algorithms, either platform-specific for multiple metamodels or specific metamodels (Fill and Karagiannis 2013). Figure 5.4 shows an example of the ADOxx Development Toolkit on the left and the ADOxx Modeling Toolkit on the right.

5.2.2.2 MetaEdit+

MetaEdit+ is an environment for Computer Aided Software Engineering (CASE) and Computer Aided Method Engineering (CAME) and was developed by Meta-Case in 1995. It provides extensive multi-user and multi-tool support. MetaEdit+

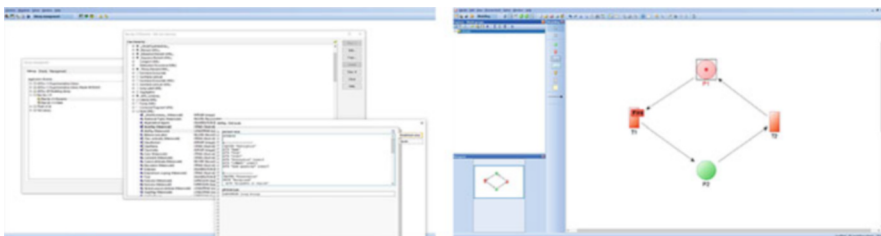


Fig. 5.4 Example of the ADOxx Development Toolkit on the left and the ADOxx Modeling Toolkit on the right. In the Development Toolkit, an example of defining the GRAPHREP for the UML metamodel for the *State* class is visible. The Modeling Toolkit, shows an example of a model instantiation of the Petri Net metamodel (Petri and Reisig 2008)

focuses on meeting the needs for flexibility, integration of methods, and representational richness. It is a multi-method, multi-user, multi-tool platform for both computer-aided software engineering and computer-aided method engineering. MetaEdit+ incorporates the same architectural principles, such as object-oriented programming, a layered database structure, and conceptual modeling (Kelly et al. 1996).

Similarly to the ADOxx Metamodeling Platform introduced in Sect. 5.2.2.1, MetaEdit+ consists of two main components. (1) The MetaEdit+ Workbench and (2) the MetaEdit+ Modeler. The MetaEdit+ Workbench is used to define metamodels and modeling methods. The MetaEdit+ Modeler is used for modeling itself, which is based on metamodels defined in the MetaEdit+ Workbench by a metamodeler.

MetaEdit+ is based on the GOPPRR meta²-model (Kelly and Tolvanen 2008). GOPPRR is an abbreviation and stands for the main concepts Graph, Object, Port, Property, Role, Relationship—see GOPPRR meta²-model in Kern et al. (2011).

5.2.2.3 GME and WebGME

The Institute for Software Integrated Systems at Vanderbilt University developed the generic modeling environment (GME) metamodeling tool (Ledeczki et al. 2001). GME is composed of two software parts. (1) MetaGME for language definition and (2) GME, a tool for language use. For language definition, the visualization of the different language concepts and the description of metamodels are considered. A figure of the GME meta²-model is available in Kern et al. (2011).

The beginning of a language is the formation of a *paradigm*. This *paradigm* is composed of *model types*. Each *model type* outlines a collection of *models*. Additionally, a *model type* can include concepts inherited from first class object (FCO). A *model* is depicted as a graphical representation on a canvas. Alternatively, a *model* can be a *symbol* on a canvas. The *symbol* in this instance serves as a reference to a model. By double-clicking on it in GME, the underlying *model* can be accessed. This concept enables a hierarchical structure of models and facilitates the refinement of models, similar to the explosion concept of MetaEdit+ mentioned in Sect. 5.2.2.2. A *model type* is composed of *atoms*, which are classes of objects that symbolize *entities* or *nodes* on a canvas and can be linked to other objects.

GME offers three ways for defining relationships between atoms. *Connection*, *reference* and *set*. A *connection* connects two *atoms*. A *connection* is like a binary association and is visually represented by a line connecting two *atoms*. A connection has roles that define which *atom* or *model* are participating in a connection. An alternative to connections is a reference between elements. This type of relationship does not have a graphical representation in a model, and is similar to INTERREF in ADOxx and the reference concept in MetaEdit+. A third approach is to establish a *set relationship*. The graphical representation of a *set relationship* enables the user to display or conceal items associated with a chosen element, similar to the decomposition and explosion concepts of MetaEdit+. Each subclass of FCO can possess *attributes*, and GME allows the inheritance of metamodeling elements (Kern 2016).

On the basis of GME, WebGME was developed. WebGME is a web- and cloud-based metamodeling tool that enables collaboration and scalability for the design of DSMLs and the creation of corresponding domain models. GME’s prototypical inheritance concepts is extended in WebGME to fusion metamodeling and modeling (Maróti et al. 2014). At any time during modeling each model is also a prototype that can be used to create an instance model.

This approach is distinct from the inheritance found in object-oriented programming languages or other modeling language approaches. It combines composition and inheritance. WebGME introduces a novel concept that blurs the distinction between metamodeling and domain modeling by using inheritance to capture the relationship between the metamodel and the model. Every model in a WebGME project is held within a single hierarchy structure with a model known as FCO at its root. Each instance inherits all constraints and rules from its parents recursively up to the FCO root and can refine it further by adding information. This is a type of multi-level metamodeling that can theoretically have an infinite amount of levels (Maróti et al. 2014).

As visible in Fig. 5.5, the architecture of WebGME is in accordance with the requirements of a metamodeling platform described above—see Fig. 5.2. The main difference from other metamodeling platforms is the fusion of the different bases into one single base, since there is no hard distinction between metamodels and model instances.

WebGME is a single-page web app built with JavaScript and running on a Node.js process. For its database, the developers chose MongoDB. The core and

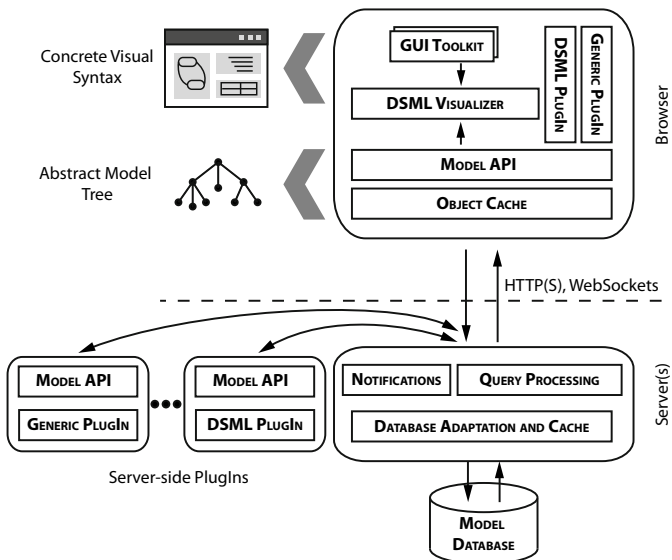


Fig. 5.5 High-level system architecture of the WebGME metamodeling platform. Reprint from Maróti et al. (2014)

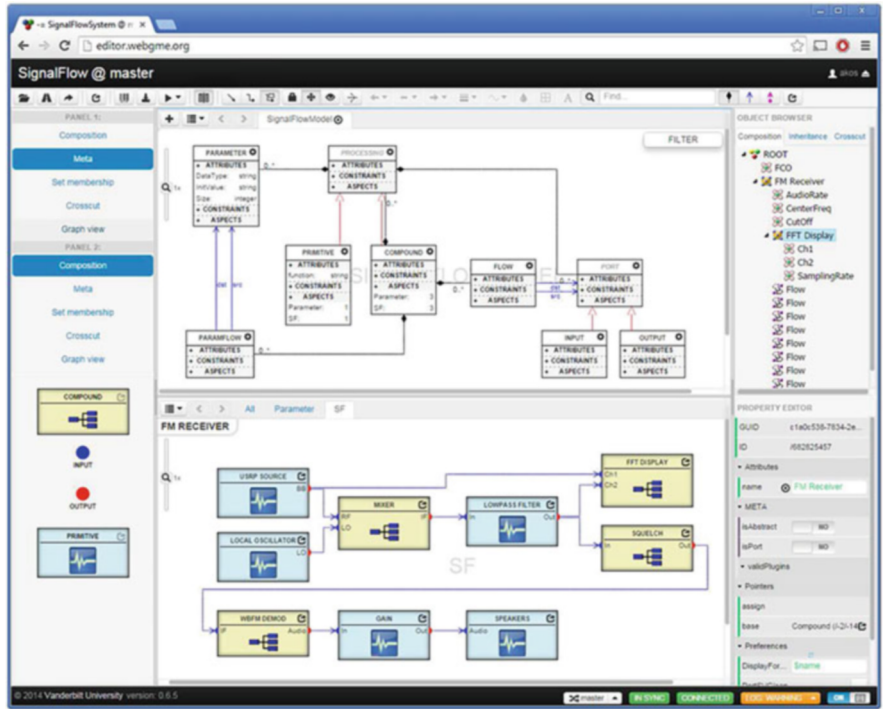


Fig. 5.6 Example of the WebGME metamodeling client combining the metamodel base with the model base. Reprint from Maróti et al. (2014)

client components supply the Model API and data access, with the Model API serving as the foundation for higher-level components such as visualization. The visualization is created using JavaScript. A REST-API enables communication between client- and the server-side applications. The JSON data format is used as standard way for importing and exporting data. Furthermore, WebGME supports multi-user modeling by implementing authentication and authorization tasks (Maróti et al. 2014). An example of the WebGME metamodeling client is visible in Fig. 5.6.

5.2.2.4 AToMPM

AToMPM stands for “A Tool for Multi-Paradigm Modeling”. It is an open-source framework for designing DSML environments, performing model transformations, and manipulating and managing models (Syriani et al. 2013). AToMPM has been developed in a collaboration of researchers at the University of Alabama (USA), the University of Antwerp (Belgium), the University of Montreal (Canada) and McGill University (Canada).

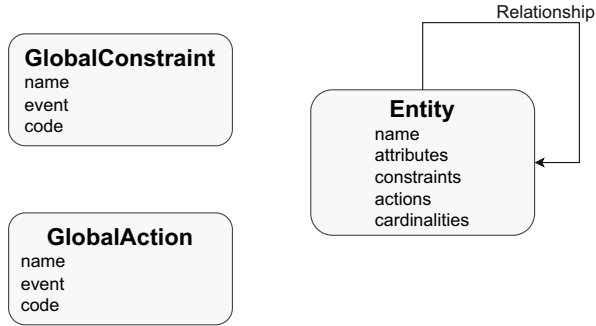


Fig. 5.7 AToMPM meta²-model. According to Mannadiar (2012)

AToMPM adheres to the idea of explicitly representing everything, at the most suitable level of abstraction, with the most suitable formalism and process, and being modeled completely by itself, i.e., AToMPM is modeled explicitly using a mixture of UML Class diagrams and statecharts. A metamodeler can also define his own modeling method to define metamodels, as long as a mapping to the default metamodel is provided (Syriani et al. 2013).

Compared to the metamodeling platforms introduced above, the meta²-model of AToMPM is simple. It is the metamodel of the *Entity Relationship Model* (Chen 1976)—see Fig. 5.7. Furthermore, there exists also an alternative meta²-model as UML class diagram (Mannadiar 2012).

AToMPM distinguished between abstract syntax and concrete syntax. The abstract syntax represents the instance data of the model, e.g., an instance with different attribute values, according to the metamodel of a given DSL. The concrete syntax defines how the model is displayed in a graphical way. The abstract syntax of a model can be linked to multiple concrete syntax definitions, thus a model instance can be visualized in different ways, depending on the knowledge and interest of a user (Mannadiar 2012; Syriani et al. 2013). Unlike most other metamodeling platforms, this separation of the graphical representation of modeling concepts from the metamodel of the modeling language allows to have a variety of different representations belonging to the same metamodel.

Stakeholders are able to share models, allowing modelers with varying skills to work collaboratively on the same model. In AToMPM, the concrete syntax is denoted as a view. A view is a way of representing a portion of an abstract model in a visual representation that is most suitable for the expert modeler working on that part of the model. It is a projection of the model onto a language that is tailored to the modeler’s needs. Each view defines a mapping of the different model elements to the concrete syntax. This is necessary since these mappings are view-specific (Corley and Syriani 2014).

AToMPM provides the user with one graphical interface for the definition of metamodels, as well as for modeling instance models. By default, AToMPM allows users to edit models, create modeling languages, define abstract and concrete syntax,

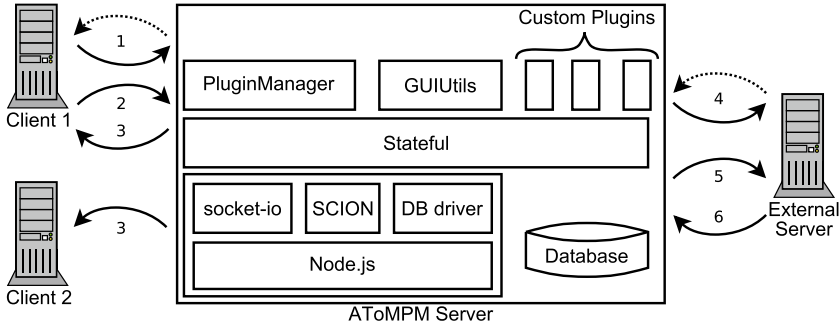


Fig. 5.8 High-level architecture of the AToMPM modeling platform. Reprint from Syriani et al. (2013)

and execute model transformations (Corley and Syriani 2014). AToMPM is a web-based modeling tool that does not require any installation on the user's computer. It can be used in the cloud, or the server can be installed on the user's premises. The only requirement for the client is a web browser that supports SVG. Models can be downloaded to the user's local device if desired (Syriani et al. 2013). Similarly to the ADOxx metamodeling platform, SVG elements are used to display all model elements. AToMPM offers a range of static and dynamic manipulation options, such as translation, scaling, rotation, transparency, and Bézier curves. In addition, a textual language can be used to perform the same manipulations as in the graphical editor by means of text commands (Syriani et al. 2013).

As depicted in Fig. 5.8, the architecture of AToMPM is split into two parts: A front-end web server, which allows multi-client connections on the same server, and a back-end server. The front-end server of AToMPM is built on Node.js and can be extended with plugins. The server and its plugins are able to communicate with each other through the use of the *State Chart Extensible Markup Language* (SCXML). The *AMS messaging system* is responsible for the communication between the web client and the server (Corley and Syriani 2014). By providing such a modular architecture, AToMPM conforms to the general structure of metamodeling platforms introduced in Fig. 5.2. Since AToMPM only provides one client application, the *Metamodel Base* and the *Model Base* are not strictly separated from each other and are denoted as *modeling and metamodeling kernel* (MMKernel) (Mannadiar 2012). An example of the AToMPM modeling client for the use case of modeling a Petri Net diagram (Petri and Reisig 2008) is depicted in Fig. 5.9.

This section does not cover all metamodeling platforms used in academia and industry. There are many more, some of them correspond to the architecture proposed by Karagiannis and Kühn (2002) for metamodeling platforms, and some do not. As this section aims to showcase some metamodeling platforms and their similarities and differences rather than providing a conclusive comparison of metamodeling platforms, interested readers are referred to Kern et al. (2011) and Kern (2016).

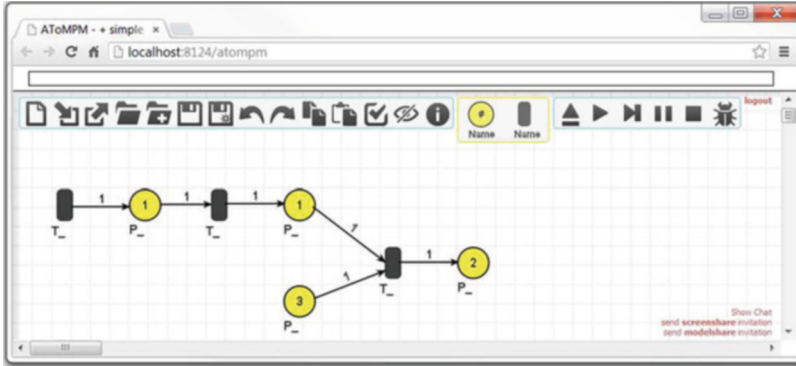


Fig. 5.9 Example of the AToMPM modeling web client, modeling a Petri Net diagram (Petri and Reisig 2008). Reprint from Syriani et al. (2013)

5.2.3 Implications for the Derivation of the Modeling Method

After evaluating various augmented reality development approaches and introducing some metamodeling platforms, we can draw conclusions about the most suitable methods in both areas in the context of this chapter.

As described in the next section, the methodology followed proposes to use a metamodeling language to derive and implement new DSMLs. We chose the ADOxx metamodeling platform for the initial implementation of the modeling language due to its extensive capabilities and the ability to quickly create visual modeling languages (see Sect. 5.2.2.1).

For the demonstration of the execution of the generated models, we will introduce an AR engine. We chose to use the WebXR device API as the development platform for our prototype because it is advantageous to base research on open-source standards that are available to everyone, cf. Sect. 5.2.1.6.

5.3 Derivation of the Visual Modeling Method

Domain-specific languages in general provide constructs that are tailored to a specific field of application with the goal of gaining expressiveness and ease of use to increase productivity (Mernik et al. 2005). In the area of model-driven software development, typically languages with a visual notation are proposed, which we will denote in the following as *domain-specific visual modeling languages*, cf. (Frank 2013; Karagiannis et al. 2016). Related to this is a trend found today in industrial software development with the rise of low-code and no-code approaches which aim to empower users to develop software with less or no programming expertise (Bock and Frank 2021; Di Ruscio et al. 2022). Thus, we will derive a domain-specific visual modeling method for creating augmented reality applications.



Fig. 5.10 Seven phases for domain-specific language development according to Frank (2013, p. 8). Adapted from Muff and Fill (2023c)

5.3.1 Methodology

Several guidelines and methodologies have been proposed for the development of domain-specific languages, cf. Karsai et al. (2009), Frank (2013), Jannaber et al. (2017), Buchmann and Karagiannis (2015), Visic et al. (2015). We will mainly follow the macro process proposed by Frank (2013), who describes seven phases, including details for each phase—see Fig. 5.10. This macro process fits well to the different phases of the *method engineering cycle* described in Buchmann et al. (2013). For the language specification and the creation of the modeling tool, we further considered the methodology by Visic et al. (2015), which focuses on the interplay between a modeling language and algorithms, and the deployment of the modeling tool. Furthermore, as the ADOxx metamodeling platform was utilized to define the modeling method, the development implicitly follows some of the *Agile Modeling Method Engineering* (AMME) work procedures outlined in Buchmann and Karagiannis (2015).

In terms of *scope and purpose*, we aim for a method that allows users without programming expertise to create augmented reality applications that include complex workflows and run in a web browser without additional plugins or software components on a wide range of devices.

5.3.2 Requirements

Frank (2013) distinguishes between *generic* and *specific* requirements that must be analyzed prior to language specification. As Gulden and Yu (2018) pointed out, these requirements have to be carefully balanced for considering trade-offs between different design alternatives, especially in terms of simplicity, comprehensibility, and convenience of use of the language (Frank 2013).

Thus, we defined the following seven **generic requirements** (\mathbf{GR}_{1-7}) for our language as proposed by Frank (2013) and in similar fashion by Karsai et al. (2009), as well as Jannaber et al. (2017): \mathbf{GR}_1 : The language should allow the specification of AR applications of various types without programming skills, making AR application development more intuitive and user-friendly than traditional approaches. \mathbf{GR}_2 : The modeling language should use concepts that a potential user is familiar with, i.e., concepts that are either common in everyday life or related to AR environments. \mathbf{GR}_3 : The modeling language should contain special

constructs that are tailored to the domain of augmented reality. These terms need to be understood in the same way in all situations and by all users. **GR₄**: The language constructs should allow modeling at a level of detail sufficient for all foreseeable **AR** applications. **GR₅**: The language should provide different levels of abstraction to avoid overloading, thus compromising the proper interpretation of a model. **GR₆**: There should be a clear association between the language constructs and the constructs of the relevant target representations in the **AR** application. **GR₇**: In addition, Frank describes the requirement of choosing an appropriate metamodeling language that is consistent with the generic requirements described, which we will consider later for the language specification.

Furthermore, we added twelve specific requirements **SR_{1–12}** that originate from: (a) Our analysis of the domain of augmented reality in the form of fundamental concepts and existing software platforms and approaches (see Sects. 2.2.2 and 5.2.1), (b) previously identified academic approaches in the area of model-driven engineering for **AR** (see Sects. 2.3 and 5.1), and (c) requirements concerning the implementation of the language in terms of satisfying the purpose of platform-independent execution using WebXR (see Sect. 5.2.1). The specific requirements have been further grouped into three categories: *Domain*, *Abstraction*, and *Implementation*.

The category *Domain* refers to specific requirements that emerge from the domain of augmented reality applications. **SR₁**: Superimposing virtual objects on the real world (*Augmentation*) is the main functionality of augmented reality applications (Ruminski and Walczak 2014; Grambow et al. 2021; Seiger et al. 2021; Lechner 2013; Campos-López et al. 2021; Ruiz-Rube et al. 2020; Wild et al. 2014). The domain-specific modeling language must allow the user to represent virtual augmentations in various forms such as images, text labels, animations, or 3D objects, cf. Sect. 4.3. **SR₂**: To create a realistic **AR** experience, the digital augmentations superimposed on the physical world must align with the real world (Campos-López et al. 2021; Ruiz-Rube et al. 2020; Wild et al. 2014). A virtual augmentation placed on a real object should remain in its original position relative to the real object, even as the user moves around. Therefore, the modeling language must provide a concept for creating a local real-world origin to provide a reference point at application run-time (*World Origin Reference*), cf. Sect. 4.2. **SR₃**: It must be possible to specify the location of virtual augmentations in relation to other objects or the world origin in real or virtual space during model specification (*Reference Point*), cf. Sect. 4.2 and Lechner (2013), Campos-López et al. (2021), Wild et al. (2014). **SR₄**: It must be possible to specify real-world objects that can be tracked during application run-time (*Detectable/Trackable*) (Ruminski and Walczak 2014; Grambow et al. 2021; Lechner 2013; Campos-López et al. 2021; Ruiz-Rube et al. 2020; Wild et al. 2014). Therefore, a concept is required to create such detectable objects during modeling. These detectables should not only specify the existence of a real-world object, but also provide data to recognize these objects at run-time, for example, using images or 3D object data, cf. Sect. 4.4. **SR₅**: Specifying the modification of different objects based on different *actions* is a critical functionality of **AR** applications (Ruminski and Walczak 2014; Seiger et al. 2021; Lechner 2013;

Ruiz-Rube et al. 2020; Wild et al. 2014). Thus, the modeling language should permit to define transitions to subsequent actions and to directly manipulate and transform augmentations. **SR₆**: For realizing complex AR workflows (Grambow et al. 2021; Seiger et al. 2021; Wild et al. 2014), *triggers and conditions* are required to enable dynamic branchings in AR applications (Ruminski and Walczak 2014; Grambow et al. 2021; Seiger et al. 2021; Campos-López et al. 2021; Ruiz-Rube et al. 2020; Wild et al. 2014).

The category *Abstraction* refers to a general aspect to create an AR modeling language and contains only one specific requirement, which details the generic requirement of different abstraction levels (**GR₅**). **SR₇**: To reduce complexity and to separate the different roles required during the specification of AR scenarios, the modeling language should include concepts for abstraction, e.g., model decomposition, and separation of concerns to allow task sharing among stakeholders with different responsibilities (Ruminski and Walczak 2014; Lechner 2013; Ruiz-Rube et al. 2020; Wild et al. 2014)—see Sect. 2.1. For example, a designer could work on visualizing augmentations, while a domain expert could specify the application workflow.

The final category, *Implementation*, considers the requirements that must be supported in terms of language specification and implementation. **SR₈**: Due to the nature of modeling languages, an abstract and a concrete syntax in textual notation needs to be provided (Frank 2013; Karsai et al. 2009), also to ease future interoperability with previous approaches (Ruminski and Walczak 2014; Grambow et al. 2021; Seiger et al. 2021; Lechner 2013; Campos-López et al. 2021; Ruiz-Rube et al. 2020; Wild et al. 2014). In addition, since visual notation is more intuitive and user-friendly than text-based notation, a two-dimensional graphical notation needs to be specified (Grambow et al. 2021). Finally, since the AR domain reverts largely to 3D content, specifying models directly in a 3D environment is useful to facilitate spatial imagination (Seiger et al. 2021; Campos-López et al. 2021; Wild et al. 2014). Therefore, a domain-specific modeling language should consider concepts for text-based, 2D visual, and 3D spatial modeling. **SR₉**: To allow an easy and rapid adaptation of the language as requirements change, the modeling language should be based on *metamodeling* (Campos-López et al. 2021; Ruiz-Rube et al. 2020; Frank 2013).

SR₁₀: It should be possible to directly feed the model into an AR application for the *execution* of the modeled AR scenario (Ruminski and Walczak 2014; Grambow et al. 2021; Ruiz-Rube et al. 2020; Wild et al. 2014). Thus, a domain-specific modeling language for AR applications should provide a data format that can be processed by an AR engine during run-time (Fill et al. 2021) or generate code for creating the AR application itself from the models (Grambow et al. 2021). **SR₁₁**: AR applications are often built using commercial SDKs such as Apple ARKit, Wikitude, or Vuforia, most of which depend on the closed-source Unity development platform. To make the modeling language widely applicable on a wide range of devices and enable non-commercial long-term research, the modeling language (*specification*) and code generated from it (*execution*) should be based on open standards. **SR₁₂**: To ensure reproducibility and accessibility, the implementation of the domain-specific

modeling language should be made openly available (Seiger et al. 2021; Ruiz-Rube et al. 2020; Wild et al. 2014).

5.3.3 Language Specification

According to Frank (2013), the phase of *language specification* contains several parts. The first step is to create a glossary containing all concepts that are considered relevant to the domain of discourse. These terms were derived from the requirements shown above, e.g., *augmentation*, *detectable*, or *condition*. Next, for each concept in the glossary, it has to be decided whether it should be part of the modeling language and how it will be expressed with the language during instantiation. Further, it needs to be decided which *metamodeling language* or *meta²-model* should be used. Subsequent to the language specification, Frank (2013) foresees a separate phase for the design of the graphical notation. First, an overview of the language concepts and the abstract syntax is presented in the form of a metamodel. Thereafter, we show the graphical notation and details on the semantics of the constructs.

For the definition of the modeling method, we used the metamodeling language of ADOxx (Fill and Karagiannis 2013)—see Sect. 5.2.2.1. ADOxx was chosen due to its wide usage within projects of the OMiLAB network (Göttinger et al. 2016) and the availability of an open platform for the implementation of model editors. The main metamodeling concepts in ADOxx are (Fill and Karagiannis 2013; Fill et al. 2012b): *ModelType* (M), *Class* (C), *Relationclass* (R), and *Attribute* (A). Modeltypes contain one or more classes, which may be connected by relationclasses. Modeltypes, classes, and relationclasses may have attributes. Instances of classes and relationclasses can only be contained in one particular instance of a modeltype. Special attributes of type **<Interref>** act as pointers to other class instances or model instances. In the metamodel introduced in the following, each concept will be marked with the icons introduced above (M), (C), (R), (A) to indicate the corresponding meta²-concept.

Figure 5.11 shows the metamodel of the new domain-specific modeling language. The modeling language is divided into three separate *ModelTypes* (M): *ObjectSpace*, *Statechange*, and *FlowScene*. This results from requirements **GR₂**, **GR₅** and **SR₇**. An *ObjectSpace* (M) defines the real world of an AR environment. It contains the two classes *Augmentation* (C) and *Detectable* (C) as defined by requirements **SR₁** and **SR₄**. Further, augmentations can include other augmentations, indicated by the *child* (R) relationclass and they may be connected to Detectables via *anchored* (R) relations (**SR₃**). A *Detectable* has an attribute *is_origin* (A), specifying if a *Detectable* references the world origin (**SR₂**).

Statechanges are described in the separate *ModelType* *Statechange* (M)—**SR₅** and **SR₇**. Within such models, Augmentations from the *ObjectSpace* model are referenced (*Reference* (C)) and changes on their attributes—e.g., a rotation transformation—are expressed via the attribute *statechange_list* (A).

The *FlowScene* (M) *ModelType* defines the workflow of the AR application and how it reacts to different environmental conditions (**GR₄**, **SR₆**). Every

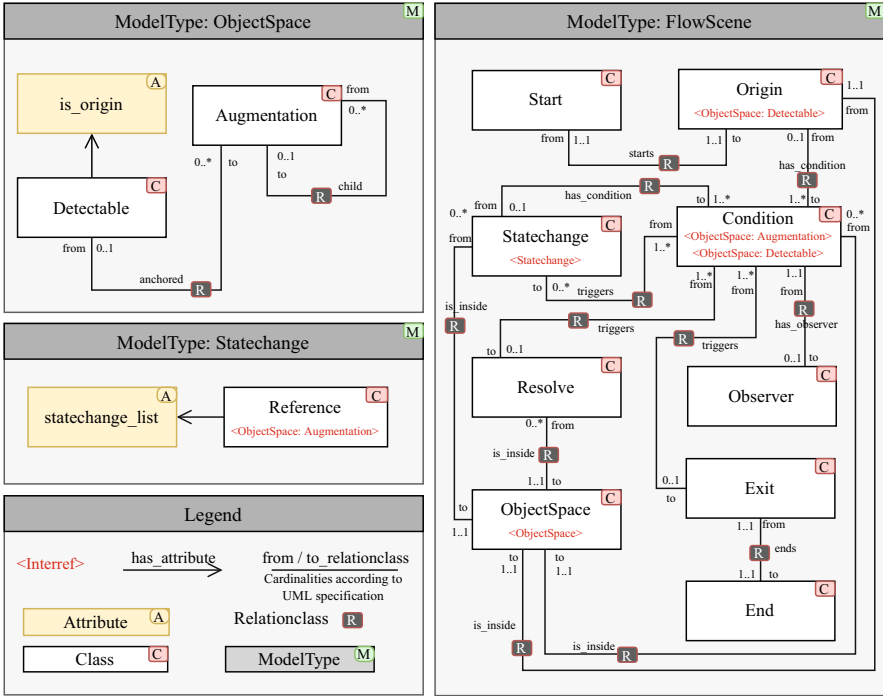

















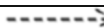



Fig. 5.11 Metamodel of the DSML for augmented reality applications with the three modeltypes ObjectSpace, Statechange, and FlowScene, as well as a legend. Adapted from Muff and Fill (2023c)

FlowScene contains exactly one *Start* (C) and one *End* (C) instance (SR₆). Each *FlowScene* contains an *ObjectSpace* (C) instance, which references an instance of the *ObjectSpace* ModelType. Inside this *ObjectSpace* class instance, the *FlowScene* model defines an *Origin* (C), one or multiple *Statechanges* (C), *Conditions* (C), and *Resolves* (C) (SR₂, SR₆). They are linked to the *ObjectSpace* with the *is_inside* (R) relationclass, specifying that these concepts are linked to one specific *ObjectSpace*. The *Origin* is used to define the world origin of the AR environment. Thus, it references a *Detectable* in the *ObjectSpace* model. *Conditions* (C) define requirements which are necessary to trigger the subsequent *Statechanges*, or to trigger *Resolves*, if there are no consecutive *Statechanges* (SR₆). Thus, *Statechanges* and *Resolves* are connected to *Conditions* by the *triggers* (R) relationclass. *Conditions*, on the other hand, follow an *Origin* or *Statechange* via the *has_condition* (R) relationclass. Furthermore, *Conditions* can be associated with an *Observer* (C) using the *has_observer* (R) relationclass. *Observers* can be used to monitor sensor data or APIs (SR₆).

For each of the classes and relationclasses, we added a graphical notation and details about the meaning of each construct in the form of a semantic definition, as shown in Table 5.2. Thereby, we considered principles from graphical

Table 5.2 Semantics and notation of the modeling language. For each ModelType, the semantic definition of the contained constructs is explained, and the visual notation is shown. Adapted from Muff and Fill (2023c)

	Concept	Semantic Definition	Notation
ObjectSpace	Detectable	Supplying configuration information to sensory processing and computer vision systems, guiding them to identify physical objects. <i>Detectables</i> are an integral component of the AR environment and may be affixed to real-world objects.	
	Augmentation	Virtual, visual, or acoustic content that is fueled into the AR environment with a given position and orientation relative to its parent <i>Augmentation</i> , a <i>Detectable</i> , or the world origin of the AR environment. Can be of the type image, animation, 3D object, video, audio, label, or link.	
	Anchored	Relationship type that allows connecting <i>Augmentations</i> with a <i>Detectable</i> . This is used to specify the position of <i>Augmentations</i> based on the position of real-world objects, independent of <i>Statechanges</i> . A <i>Detectable</i> can have multiple anchored <i>Augmentations</i> , but an <i>Augmentation</i> can be anchored to a maximum of one detectable.	
	Child	Relationship type used for the hierarchical structuring of <i>Augmentations</i> . An <i>Augmentation</i> can have multiple children, which in turn can have children. Useful for specifying the transformation of multiple <i>Augmentations</i> , based on a common point.	
Statechange	Reference	<i>Reference</i> is the only class of the <i>Statechange</i> ModelType. It is used to define a transformation of an <i>Augmentation</i> at a given state. It references an <i>Augmentation</i> in the <i>ObjectSpace</i> model and specifies the <i>Augmentation's</i> position, rotation, and visibility at the time of this particular <i>Statechange</i> .	
FlowScene	ObjectSpace	<i>ObjectSpace</i> is a part of the <i>FlowScene</i> model. It points to an instance of an <i>ObjectSpace</i> model. Each <i>ObjectSpace</i> instance can contain <i>Condition</i> , <i>Statechange</i> , and <i>Resolve</i> instances. All contained instances are dependent on the referenced <i>ObjectSpace</i> model.	
	Start	Indicates the start of a <i>FlowScene</i> model. There can be only one <i>Start</i> object in a model.	
	End	Indicates the end of a <i>FlowScene</i> model. There can be only one <i>End</i> object in a model.	
	Statechange	Defines a <i>Statechange</i> in the AR environment at a given point in time. <i>Statechanges</i> are triggered by <i>Conditions</i> . A <i>Statechange</i> instance references a <i>Statechange</i> model that specifies transformations of <i>Augmentations</i> at that given <i>Statechange</i> . A <i>Statechange</i> is followed again by a <i>Condition</i> . The icon (S) represents a reference to a <i>Statechange</i> .	
	Origin	Defines the world origin of the AR environment. An <i>Origin</i> depends on an instance of an <i>ObjectSpace</i> model. It must be placed on the border of an <i>ObjectSpace</i> instance and references a <i>Detectable</i> in the <i>ObjectSpace</i> model on which it depends. The <i>Origin</i> always follows a <i>Start</i> instance and is followed by one or more <i>Conditions</i> that are triggered when the referenced <i>Detectable</i> is detected in the AR environment. The icon (+) represents a reference to an <i>ObjectSpace</i> model instance.	
	Condition	Defines what <i>Condition</i> must be met to move to the next instance, which can be a <i>Statechange</i> , a <i>Resolve</i> , or an <i>Exit</i> instance. There are four types of <i>Conditions</i> , including user-driven actions (click and voice condition), visibility of <i>Detectables</i> (detect condition), conditions driven by <i>Observers</i> (observer condition), e.g., based on sensor data, and time conditions (timer condition). A <i>Condition</i> can follow multiple preceding instances of <i>Origin</i> and <i>Statechange</i> , and can have multiple subsequent instances of <i>Statechange</i> , <i>Resolve</i> , or <i>Exit</i> . To show the reference between a <i>Detectable</i> or an <i>Augmentation</i> (object) and its corresponding instance, icons (D) and (O) are used next to the triangle.	
	Resolve	Resolves an open sequence of <i>Statechanges</i> . Since it is possible to have multiple parallel sequences of <i>Statechanges</i> , it is possible to resolve a sequence without using an <i>Exit</i> instance, thus exiting the entire model. A <i>Resolve</i> instance can follow multiple <i>Conditions</i> and has no succeeding instances.	
	Observer	Additional conditional information, always being attached to a <i>condition</i> instance. <i>Observers</i> specify an observer call that can return a result at runtime. For example, an <i>observer</i> can monitor a temperature sensor and trigger a <i>condition</i> at a certain threshold.	
	Exit	<i>Exit</i> depends on an <i>ObjectSpace</i> and must be placed on the border of an <i>ObjectSpace</i> instance. It indicates that the sequences specified in the <i>ObjectSpace</i> have ended. An <i>Exit</i> instance can follow several <i>Condition</i> instances. It is always followed by exactly one <i>End</i> instance.	
	Starts	Relationship type for the entry of an <i>ObjectSpace</i> instance by an <i>Origin</i> instance. There is always exactly one <i>Starts</i> relation.	
	Has Condition	Relationship type to enter a <i>Condition</i> instance. A <i>Has Condition</i> relation can connect an <i>Origin</i> or a <i>Statechange</i> instance to a <i>Condition</i> instance.	
	Triggers	Relationship type used to trigger an action after a <i>Condition</i> is satisfied. A <i>Triggers</i> relationship can connect a <i>Condition</i> instance to a <i>Statechange</i> instance, a <i>Resolve</i> instance, an <i>Exit</i> instance, or another <i>Condition</i> .	
	Has Observer	Relationship type to connect a <i>Condition</i> instance to an <i>Observer</i> instance.	
	Ends	Relationship type for the exit of an <i>ObjectSpace</i> instance by an <i>Exit</i> instance. There is always exactly one <i>Ends</i> relation.	

notation design by Moody (2009) as far as possible. In particular, we aimed for *Semiotic Clarity*, *Perceptual Discriminability*, *Semantic Transparency*, *Complexity Management*, *Cognitive Integration*, *Visual Expressiveness*, *Dual Coding*, *Graphic Economy*, and *Cognitive Fit*. The further development of the graphical notation including more advanced methods such as recently described by Bork and Roelens (2021) is planned for the future.

5.3.4 Modeling Procedure

The modeling method's procedure is not strictly defined. As described in **SR**₇, the concept of separation of concerns allows for the independent modeling of certain parts of the language. For instance, a process specialist can define only the *FlowScene* of a model, while a content creator can define only the visualization of *Statechanges*. To ensure a seamless modeling process, we propose following the modeling procedure described below.

First, an empty *ObjectSpace* model and an empty *FlowScene* model should be created. Second, it is possible to design the workflow of the *FlowScene* model, including the instantiation of *Start* and *End*, the *ObjectSpace*, and its connection to the *ObjectSpace* model. The **AR** application's general workflow can be created by instantiating empty placeholders for *Conditions* and *Statechanges*. Third, the mapping to the real world should be modeled in the *ObjectSpace*. This means that the origin *Detectable*, as well as all other *Detectables*, should be instantiated in the *ObjectSpace*. Additionally, *Augmentations* can be created. If a content creator is performing this task, it may also be possible to create *Augmentations* in the next step. Once all necessary *Augmentations* have been created, different *Statechange* models can be created in the fourth step. For each *Statechange* in the *FlowScene*, a corresponding *Statechange* model should be created. In each *Statechange* model, *References* to the *Augmentations* to be modified in this *Statechange* should be added. For each *Reference*, the attributes for changing the properties of the referenced *Augmentation* can be modified. In the fifth step, the *FlowScene* requires the addition of the *Origin*, which should be linked to the *ObjectSpace* model. Additionally, the defined *Statechange* instances can be linked to the created *Statechange* models, and *Conditions* can be linked to *Detectables* if necessary.

It should be noted that this procedure is not strict and can be varied by experienced modelers. Nevertheless, these steps can help to ensure the creation of a valid model.

5.3.5 Mechanisms and Algorithms

The modeling language described above is not reliant on any particular mechanisms or algorithms (see Sect. 2.1.3.1). However, since the modeling method produces

different models that will be further processed for **AR** application generation, it is necessary to algorithmically process the generated models. As the methodology proposed by Frank (2013) suggests the use of a metamodeling language (**GR**₇), we assume that generic algorithms for model processing, e.g. model export, are available at this level.

5.3.6 First Implementation and Execution on ADOxx

After the language specification, the modeling language has been implemented using the freely available and open ADOxx metamodeling platform and will be made available via Zenodo (Muff and Fill 2023b). This platform was selected for its widespread availability, extensive documentation, and ability to quickly and iteratively create visual prototypes for modeling languages. The platform easily enables the definition and customization of metamodels using the ADOxx meta²-model and generation of model instances in automatically created model editors (**SR**₉). ADOxx provides several text-based formats for defining metamodels and models, as well as a DSL for graphical notation (**SR**₈)—see the GRAPHREP example in Fig. 4.7. In this way, the models can be exported manually or programmatically in XML format for processing them in other applications.

The ADOxx XML interface has been chosen as a basis to enable the execution of the modeling language (**SR**₁₀). For this purpose, a software component has been designed in the form of an **AR** engine to interpret the models. The engine is implemented as a platform-independent web application using the **3D** JavaScript library *THREE.js*²⁶ and the **VR/AR** immersive web standard *WebXR* (Jones et al. 2023). The application can be accessed through a WebXR-compatible web browser on any mobile device, such as smartphones or *head-mounted displays* in line with requirement **SR**₁₁. For starting an **AR** experience, the engine processes the models selected by the user and monitors the user's environment for potentially relevant changes. Based on these environmental changes and user interactions, the application adapts the environment according to the specified workflows specified through triggers, conditions, and actions (**SR**₆).

5.4 Evaluation of the Visual Modeling Method on ADOxx

The evaluation of the initial implementation of the visual modeling method on the ADOxx modeling platform will be presented in three parts. Firstly, Sect. 5.4.1 will show a use case of a practical application of the new visual modeling method. Next, Sect. 5.4.2 will evaluate the introduced approach against previous approaches.

²⁶ <https://github.com/mrdoob/three.js/> last visited on: 01.03.2024.

Finally, in Sect. 5.4.3, the consistency of the modeling method will be evaluated by formally describing it with the FDMM formalism (Fill et al. 2013, 2012a,b).

5.4.1 Use Case for the Domain-Specific Visual Modeling Method

We have developed a use case that demonstrates the practical application of the modeling method. The use case involves the assembly of a bedside table with the assistance of augmented reality. The objective of this use case is to assist users in assembling a bedside table using an augmented reality application instead of traditional 2D instructions on paper. Figure 5.12 displays a screenshot of the implementation in ADOxx, which includes an excerpt of a *FlowScene* model (1), the referenced *ObjectSpace* model (2), and two *Statechange* models (3 & 4).

The upper part of Fig. 5.12 displays an excerpt of the *FlowScene* model, demonstrating how to assemble the furniture piece step by step. The process involves turning the pieces into the correct position and attaching them one by one. It is important to note that the *FlowScene* model defines trigger-condition-action sequences rather than static flows. The *FlowScene* model refers to one *ObjectSpace* model (2) and several *Statechange* models (3 & 4).

The lower left part of Fig. 5.12 shows the *ObjectSpace* model (2). It contains ten *Detectables*, which contain images of markers that are well suited for computer vision detection algorithms. These act as surrogates for more advanced 3D object recognition algorithms that would allow direct detection of physical objects (cf. Sect. 4.4). Furthermore, the model includes *Augmentation* instances for each part of the furniture piece, e.g., “TopPlate 1”. These *Augmentations* are provided as GLTF files, which is a common format for 3D objects and their textures—see Sect. 4.3.5. The *Augmentations* are connected by *is_child* relations to facilitate positioning and can be assigned *Detectables* to use them as reference points by *anchored* relations. The *Augmentations* and *Detectables* defined in the *ObjectSpace* model are then referenced in the *FlowScene* model.

Furthermore, the *FlowScene* model (1) includes *Statechange* instances—e.g., “Init MiddlePlate”—which reference *Statechange* models. In the lower right of Fig. 5.12, two examples of *Statechange* models “Init MiddlePlate” (3) and “Leg 1 Positioned” (4) are shown. They reference one or more *Augmentations* from the *ObjectSpace* model and define the state of the position, rotation, and visibility parameters during the execution of the *FlowScene* model. These parameters are also displayed as a table. A detailed description of the semantics and notation of each language concept is available in Table 5.2.

The execution of the models of the use case is shown in Fig. 5.13 by using parts from an IKEA table (IKEA 2023). Subfigures (a)–(c) illustrate the traditional 2D assembly instructions for (a) “attaching Leg 1”, (b) “turning MiddlePlate 90° counterclockwise”, and (c) “attaching Leg 2”. Subfigures (d)–(f) illustrate the same

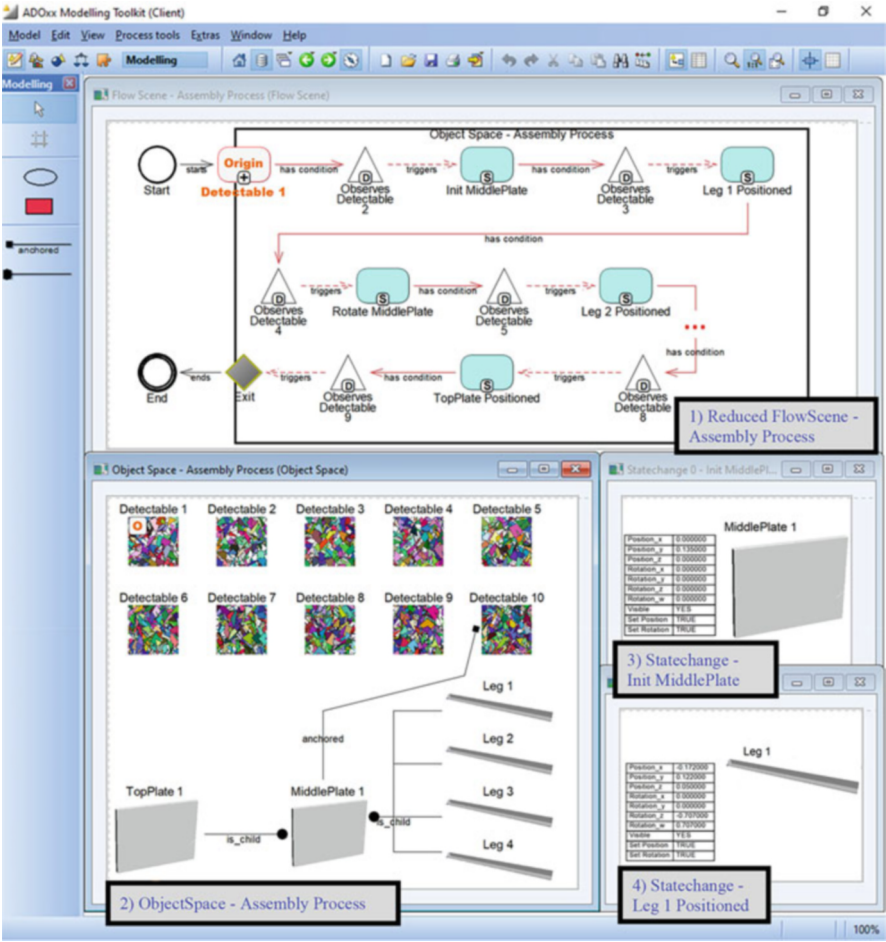


Fig. 5.12 Screenshot of the ADOxx implementation showing model excerpts for supporting an assembly process in augmented reality: (1) *FlowScene* model of the assembly process. (2) *ObjectSpace* model of the necessary augmentations and detectables using markers. (3) and (4) showing two exemplary *Statechange* models. Adapted from Muff and Fill (2023c)

steps of the instructions in augmented reality using the aforementioned models and the WebXR AR engine. The screenshots were taken while using the WebXR AR engine in the Chrome browser on a *Samsung Galaxy Tab S7* tablet. Subfigure (d) shows the *Statechange* “Leg 1 Positioned”. It superimposes an image of *Leg 1* on top of the real *MiddlePlate*, whose existence, position, and orientation are detected via a marker—*Detectable 10*. The *Statechange* “Rotate MiddlePlate”, where the virtual object is rotated according to the desired position for further assembly of the table, is shown in Subfigure (e). Subfigure (f) shows the *Statechange* “Leg 2 Positioned”. The augmentation shows where the next leg should be attached. As

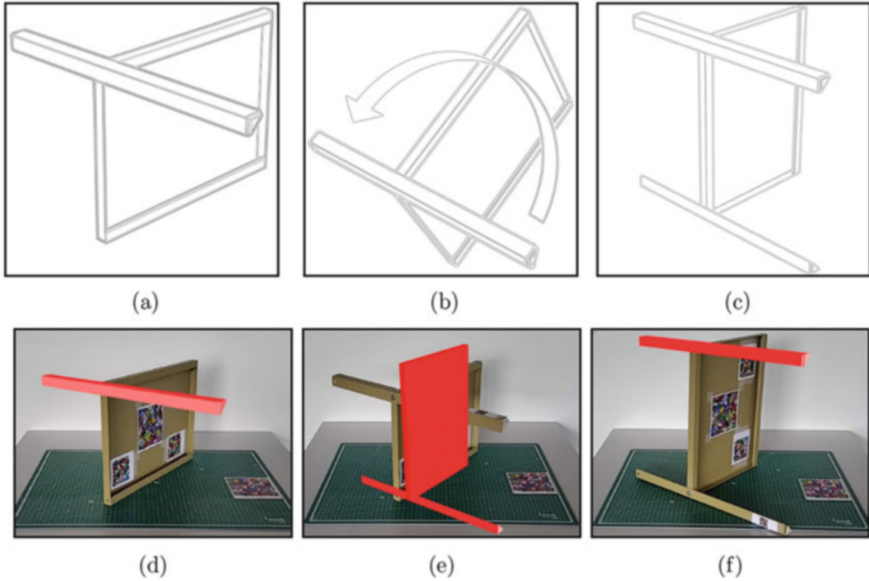


Fig. 5.13 Illustration of the assembly process of a bedside table—cf. IKEA (2023) (a–c), and the support through AR based on the visual models (d–f). Adapted from Muff and Fill (2023c)

can be seen in subfigures (d), (e) and (f), several colored markers are placed on the real object at strategic points and according to the *ObjectSpace* model. Once a marker is detected, it is decided based on the current state of the workflow defined by the *FlowScene* model if it triggers an action or not. If an action is triggered, the workflow moves on and waits until the next *Detectable* (marker) in line is detected. The flexible structure of the DSML allows multiple workflow paths to be active at the same time by simultaneously checking for multiple detectables. Detectables are also tracked when they are not part of the *FlowScene*. To avoid making the use case unnecessary complex, the concepts of *Resolves* and *Observer* were not used.

5.4.2 Comparative Evaluation of the First Prototype

Several techniques can be chosen to evaluate the new modeling method denoted as ARWFMM, including feature comparisons, theoretical and conceptual investigations, and empirical evaluations (Siau and Rossi 2011). Thereby we opted for a feature comparison to previous approaches along the specific requirements that we had formulated. The previous approaches we considered were the ones from Ruminski and Walczak (2014), Grambow et al. (2021), Seiger et al. (2021), Lechner (2013), Campos-López et al. (2021), Ruiz-Rube et al. (2020), and Wild et al. (2014).

Table 5.3 Feature comparison of the new domain-specific visual modeling method [ARWFMM](#) based on twelve specific requirements SR_1-12 . (Y): Requirement met. (N): Requirement not met. (-): Not specified. Adapted from Muff and Fill (2023c)

Approach		Ruminski & Walczak 2014	Grambow et al. 2021	Seiger et al. 2021	Lechner 2013	Campos-Lopez et al. 2021	Ruiz-Rube et al. 2020	Wild et al. 2014	ARWFMM
Domain	SR1: Augmentation								
	Animations	N	N	N	Y	N	N	Y	N
	Images	N	Y	N	Y	Y	Y	Y	Y
	Videos	N	Y	N	Y	Y	N	Y	Y
	Audio	N	N	N	Y	N	N	Y	Y
	Labels	Y	Y	Y	Y	Y	Y	Y	Y
	3D Object	Y	Y	Y	Y	Y	Y	Y	Y
	Link	N	N	Y	Y	Y	Y	N	N
	Checklist	N	Y	N	N	N	N	N	N
	Form	N	Y	N	N	N	N	N	N
	SR2: World Origin Reference	N	N	N	N	Y	Y	Y	Y
	SR3: Reference Point	N	N	N	Y	Y	N	Y	Y
	SR4: Detectables / Trackables								
	Anchor	N	N	-	N	Y	N	Y	Y
	Marker / Image	Y	Y	-	Y	Y	Y	Y	Y
	3D Object	N	N	-	Y	N	N	N	Y
	SR5: Action	Y	N	Y	Y	N	Y	Y	Y
	SR6: Triggers and Conditions								
	Click	Y	-	Y	-	Y	Y	Y	Y
	Detect	N	-	N	-	Y	Y	Y	Y
	Sensor	N	-	Y	-	N	Y	Y	Y
	Voice	N	-	N	-	N	Y	Y	Y
	Timer	N	-	N	-	Y	N	N	Y
	Area	Y	-	N	-	N	N	N	N
	Gesture	N	-	Y	-	Y	Y	N	Y
	Workflow	N	Y	Y	N	N	N	Y	Y
Abstraction	SR7: Levels of Abstraction								
	Decomposition	N	N	N	N	N	Y	Y	Y
	Separation of Concerns	Y	N	N	Y	N	Y	N	Y
Implementation	SR8: User Interaction								
	Text-based Modeling	Y	Y	Y	Y	Y	Y	Y	Y
	2D Visual Modeling	N	Y	N	N	N	N	N	Y
	3D Spatial Modeling	N	N	Y	N	Y	N	Y	N
	SR9: Metamodeling	N	N	N	N	Y	Y	N	Y
	SR10: Model Execution	Y	Y	N	-	N	Y	Y	Y
	SR11: Open 3D Standard Support								
	Specification	N	N	N	N	N	N	N	N
	Execution	N	N	N	N	N	N	N	Y
	SR12: Openly Available	N	N	Y	N	N	Y	Y	Y
Σ of supported requirements		9	11	11	13	16	18	21	26

For each specific requirement that we had formulated, we conducted a detailed comparison using multiple dimensions, as shown in Table 5.3. This provides a detailed overview of the features supported by previous approaches and our new modeling method in terms of augmented reality concepts, levels of abstraction, user interaction, metamodeling capabilities, model execution, support for open standards, and availability of corresponding implementations. Thus, we can show that our new modeling method [ARWFMM](#) currently supports 26 out of 33 dimensions of requirements, while the next runner-up only supports 21 dimensions.

In regard to *Augmentations* (**SR**₁), features such as animations, links, checklists, and forms are not yet supported by our language. This is more of a technical than a conceptual issue and will be addressed in future versions. The same holds true for area triggers (**SR**₆). Concerning *User Interaction* (**SR**₈), the current implementation of our language only supports text-based and 2D visual modeling, which is due to limitations of the ADOxx platform, which is not yet available as open source. 3D spatial modeling, such as in a 3D-capable modeling tool or directly in AR, is not yet supported. For enabling 3D spatial modeling, the adaptation of current metamodeling platforms would be necessary, e.g., for directly supporting open 3D standards such as *WebXR* (**SR**₁₁). This would certainly facilitate the specification of models, as 3D modeling greatly facilitates spatial imagination.

5.4.3 Formal Evaluation of the Domain-Specific Modeling Language

In addition to the comparative evaluation in Sect. 5.4.2, we also evaluated the DSML in a more formal way with the FDMM formalism. FDMM aims to provide a user-friendly formalism that does not require specialized mathematical knowledge and is capable of expressing the implementation of different metamodels and model instances (Fill et al. 2012b). The advantage of such a formalization is that the consistency of a modeling language can be shown mathematically, independent of a specific modeling platform. Another advantage of FDMM is that it can be used to describe algorithms on models independently of the implementation, e.g., for transformation or queries (Pittl and Fill 2020; Johannsen and Fill 2017). First, we define the basic components of the metamodels, which can be used to derive model instances. A metamodel is defined as a tuple in the form of:

$$\mathbf{MM} = \langle MT, \preceq, domain, range, card \rangle \quad (5.1)$$

where MT is a set of model types. Thus, there are:

$$\mathbf{MT} = \{MT_1, MT_2, \dots, MT_n\} \quad (5.2)$$

where MT_i is a tuple:

$$\mathbf{MT}_i = \langle \mathbf{O}_i^T, \mathbf{D}_i^T, \mathbf{A}_i \rangle \quad (5.3)$$

Thereby, a tuple consists of object types \mathbf{O}^T , a set of data types \mathbf{D}^T and a set of attributes \mathbf{A}_i . In addition, attributes are used to describe associations between object types. For readers interested in more details on the formal definition of the FDMM formalism, we refer to Fill et al. (2013, 2012a,b). In the following, the metamodel of the ARWFMM modeling language, introduced in Fig. 5.11, is evaluated by

formalizing it with the FDMM formalism, thus demonstrating the consistency of the metamodel and its connections.

5.4.3.1 Modeltypes

The **ARWFMM** modeling method consists of one metamodel consisting of three *ModelTypes*. The *ObjectSpace*, *Statechange*, and *FlowScene*, which can thus be formalized as three separate model types (MT) as follows:

$$\mathbf{MT}_{\text{ObjectSpace-Model-Type}} = \left\langle \mathbf{O}_{\text{ObjectSpace-Model-Type}}^T, \mathbf{D}_{\text{ObjectSpace-Model-Type}}^T, \mathbf{A}_{\text{ObjectSpace-Model-Type}} \right\rangle \quad (5.4)$$

$$\mathbf{MT}_{\text{Statechange-Model-Type}} = \left\langle \mathbf{O}_{\text{Statechange-Model-Type}}^T, \mathbf{D}_{\text{Statechange-Model-Type}}^T, \mathbf{A}_{\text{Statechange-Model-Type}} \right\rangle \quad (5.5)$$

$$\mathbf{MT}_{\text{FlowScene-Model-Type}} = \left\langle \mathbf{O}_{\text{FlowScene-Model-Type}}^T, \mathbf{D}_{\text{FlowScene-Model-Type}}^T, \mathbf{A}_{\text{FlowScene-Model-Type}} \right\rangle \quad (5.6)$$

Each of these three model types has its own object types \mathbf{O}^T , a set of data types \mathbf{D}^T and a set of attributes \mathbf{A}_i , which will be introduced in the following.

5.4.3.2 Object Types

Each *class* and *relationclass* in an *ADOxx metamodel* is expressed as an object type. Thus, there are a total of 20 object types in the three object type sets for *ObjectSpace*, *Statechange*, and *FlowScene*.

$$\mathbf{O}_{\text{ObjectSpace-Model-Type}}^T = \{\text{anchored, Augmentation, child, Detectable, AR_Modeling}\} \quad (5.7)$$

$$\mathbf{O}_{\text{Statechange-Model-Type}}^T = \{\text{Reference, AR_Modeling}\} \quad (5.8)$$

$$\begin{aligned} \mathbf{O}_{\text{FlowScene-Model-Type}}^T = \{ & \text{Condition, End, Exit, ObjectSpace, Observer, Origin,} \\ & \text{Start, Resolve, Statechange, is_inside, triggers, has_condition,} \\ & \text{has_observer, ends, starts, AR_Modeling} \} \end{aligned} \quad (5.9)$$

Thereby, the object type *AR_Modeling* is defined as abstract. In the ADOxx metamodel implementation, this is an abstract superclass.

5.4.3.3 Data Types

FDMM formalized the different data types available in a model type. There are a total of six different data types used in the three model types *ObjectSpace*, *Statechange*, and *FlowScene*. It should be noted that these data types are not defined by FDMM but rather by the modeling platform used to implement the metamodels. Therefore, in this case, they are ADOxx data types.

$$\mathbf{D}_{\text{ObjectSpace-Model-Type}}^T = \{ \text{ENUMERATION}, \text{DOUBLE}, \text{STRING}, \\ \text{PROGRAMCALL}, \text{CLOB} \} \quad (5.10)$$

$$\mathbf{D}_{\text{Statechange-Model-Type}}^T = \{ \text{ENUMERATION}, \text{DOUBLE}, \text{STRING} \} \quad (5.11)$$

$$\mathbf{D}_{\text{FlowScene-Model-Type}}^T = \{ \text{ENUMERATION}, \text{INTEGER}, \text{STRING} \} \quad (5.12)$$

5.4.3.4 Attribute Definitions

FDMM formalized the different attribute sets for the object types of the different model types. The three model types, *ObjectSpace*, *Statechange*, and *FlowScene*, contain 16, 12, and 20 attributes, respectively.

$$\mathbf{A}_{\text{ObjectSpace-Model-Type}} = \{ \text{is_origin}, \text{base64}, \text{Externalgraphic}, \text{Object}, \\ \text{Sizeinmeters}, \text{Name}, \text{child_from}, \text{child_to}, \\ \text{ExternalAudio}, \text{ExternalVideo}, \text{gltf}, \text{Label}, \\ \text{ObjectUpload}, \text{Type}, \text{anchored_from}, \text{anchored_to} \} \quad (5.13)$$

$$\mathbf{A}_{\text{Statechange-Model-Type}} = \{ \text{Visible}, \text{SetRotation}, \text{SetPosition}, \text{Position_x}, \\ \text{Position_y}, \text{Position_z}, \text{Rotation_x}, \text{Rotation_y}, \\ \text{Rotation_z}, \text{Rotation_w}, \text{Augmentation}, \text{Name} \} \quad (5.14)$$

$$\begin{aligned}
 \mathbf{A}_{\text{FlowScene-Model-Type}} = \{ & \text{triggers_from}, \text{triggers_to}, \text{ends_from}, \\
 & \text{ends_to}, \text{starts_from}, \text{starts_to}, \text{has_obserer_from}, \\
 & \text{has_observer_to}, \text{is_inside_from}, \text{is_inside_to}, \\
 & \text{Statechange}, \text{Name}, \text{Detectable}, \text{ObserverCall}, \\
 & \text{ObserverResult}, \text{ObjectSpace}, \text{ConditionType}, \\
 & \text{ClickedObject}, \text{Timer}, \text{VoiceText} \}
 \end{aligned} \tag{5.15}$$

5.4.3.5 Subtype Definitions

The object type *AR_Modeling* is an abstract object type. The following object types are subtypes of the *AR_Modeling* object type.

$$\begin{aligned}
 \text{Augmentation} &\leq \text{AR_Modeling}, & \text{Condition} &\leq \text{AR_Modeling}, \\
 &(5.16) & &(5.22)
 \end{aligned}$$

$$\begin{aligned}
 \text{Reference} &\leq \text{AR_Modeling}, & \text{Statechange} &\leq \text{AR_Modeling}, \\
 &(5.17) & &(5.23)
 \end{aligned}$$

$$\begin{aligned}
 \text{Start} &\leq \text{AR_Modeling}, & \text{ObjectSpace} &\leq \text{AR_Modeling}, \\
 &(5.18) & &(5.24)
 \end{aligned}$$

$$\begin{aligned}
 \text{Origin} &\leq \text{AR_Modeling}, & \text{Exit} &\leq \text{AR_Modeling}, \\
 &(5.19) & &(5.25)
 \end{aligned}$$

$$\begin{aligned}
 \text{Resolve} &\leq \text{AR_Modeling}, & \text{End} &\leq \text{AR_Modeling}, \\
 &(5.20) & &(5.26)
 \end{aligned}$$

$$\begin{aligned}
 \text{Observer} &\leq \text{AR_Modeling}, & \text{Detectable} &\leq \text{AR_Modeling} \\
 &(5.21) & &(5.27)
 \end{aligned}$$

5.4.3.6 Domain, Range, and Cardinality Definitions

Having defined all the model types, object types, data types, and attribute definitions, these concepts can now be set in relation to each other via the domain, range, and cardinality functions. The domain function maps attributes to the power set of all object types. It will restrict the objects to which an attribute can map in model instances. Thus, the domain function assigns attributes to a particular set of object types. The range function assigns an attribute to the power set of all pairs of object types and model types, all data types, and all model types. This can be the relation to (1) object types in a specific model type, (2) data types, or (3) model types. In ADOxx this would refer to (1) *relationclasses* and *INTERREFs* to object types, i.e., *classes* and *relationclasses*, (2) attributes, and (3) *INTERREFs* to model types, i.e.,

to *ModelTypes*. Finally, the card function maps pairs of object types and attributes to pairs of integers, defining the minimum and maximum possible assignments (Fill et al. 2012b). Below, some examples of domain, range, card triples are visible. For a complete overview, refer to Appendix A.

$$\text{domain}(\text{child_from}) = \{\text{child}\} \quad (5.28)$$

$$\text{range}(\text{child_from}) = \{(\text{Augmentation}, \mathbf{MT}_{\text{ObjectSpace-Model-Type}})\} \quad (5.29)$$

$$\text{card}(\text{child}, \text{child_from}) = \langle 1, 1 \rangle \quad (5.30)$$

$$\text{domain}(\text{child_to}) = \{\text{child}\} \quad (5.31)$$

$$\text{range}(\text{child_to}) = \{(\text{Augmentation}, \mathbf{MT}_{\text{ObjectSpace-Model-Type}})\} \quad (5.32)$$

$$\text{card}(\text{child}, \text{child_to}) = \langle 1, 1 \rangle \quad (5.33)$$

$$\text{domain}(\text{anchored_from}) = \{\text{anchored}\} \quad (5.34)$$

$$\text{range}(\text{anchored_from}) = \{(\text{Detectable}, \mathbf{MT}_{\text{ObjectSpace-Model-Type}})\} \quad (5.35)$$

$$\text{card}(\text{anchored}, \text{anchored_from}) = \langle 1, 1 \rangle \quad (5.36)$$

5.4.3.7 Instantiation of Model Types for the Assembly Use Case

As shown in Sects. 5.4.3.1–5.4.3.6, we were able to formalize the entire ARWFMM modeling method according to the FDMM formalism. In order to show that the formalized modeling method is really consistent, we instantiated the assembly use case (see Sect. 5.4.1) in FDMM. Since the whole example is very large, we only show some parts of it in this section. For a complete overview, refer to Appendix A.

The assembly process model introduced in Sect. 5.12 consists of a total of nine models, which are denoted as **mt** in the following. There is one instance of the *FlowScene* model type, one instance of the *ObjectSpace* model type, and seven instances of the *Statechange* model type.

$$\mu_{\mathbf{MT}}(\mathbf{MT}_{\text{FlowScene-Model-Type}}) = \{\mathbf{mt}_{\text{FlowScene-AssemblyProcess}}\} \quad (5.37)$$

$$\mu_{\mathbf{MT}}(\mathbf{MT}_{\text{ObjectSpace-Model-Type}}) = \{\mathbf{mt}_{\text{ObjectSpace-AssemblyProcess}}\} \quad (5.38)$$

$$\begin{aligned} \mu_{\mathbf{MT}}(\mathbf{MT}_{\text{Statechange-Model-Type}}) = \{ \\ \mathbf{mt}_{\text{InitMiddlePlate}}, \mathbf{mt}_{\text{Leg1Positioned}}, \mathbf{mt}_{\text{RotateMiddlePlate}}, \mathbf{mt}_{\text{Leg2Positioned}}, \\ \mathbf{mt}_{\text{RotateMiddlePlate2}}, \mathbf{mt}_{\text{Leg3and4Positioned}}, \mathbf{mt}_{\text{TopPlatePositioned}} \} \end{aligned} \quad (5.39)$$

5.4.3.8 Instantiation of Object Types for the Assembly Use Case

Each instance of a *class* and *relationclass* in ADOxx corresponds to an instance of an object type in FDMM. The instantiation of object types is demonstrated by the following expressions.

$$\mu_{\mathbf{O}}(\textit{Start}, \mathbf{MT}_{\textit{FlowScene-Model-Type}}) = \{\textit{Start}\} \quad (5.40)$$

$$\mu_{\mathbf{O}}(\textit{End}, \mathbf{MT}_{\textit{FlowScene-Model-Type}}) = \{\textit{End}\} \quad (5.41)$$

$$\begin{aligned} \mu_{\mathbf{O}}(\textit{ObjectSpace}, \mathbf{MT}_{\textit{FlowScene-Model-Type}}) = \\ \{\textit{ObjectSpace} - \textit{Assembly Process}\} \end{aligned} \quad (5.42)$$

5.4.3.9 Instantiation of Data Types for the Assembly Use Case

After instantiation of the object types, the data types can be instantiated. This is basically the definition of all data values for each data type. Below are some examples of data type instantiation:

$$\mu_{\mathbf{D}}(\textit{CLOB}) = \{\textit{'ewogJhc3...KgICJ2'}, \dots\} \quad (5.43)$$

$$\mu_{\mathbf{D}}(\textit{PROGRAMCALL}) = \{\textit{'1.png'}, \textit{TopPlate.glTF}, \dots\} \quad (5.44)$$

$$\mu_{\mathbf{D}}(\textit{STRING}) = \{\textit{'Start'}, \textit{'ObjectSpace - Assembly Process'}, \dots\} \quad (5.45)$$

$$\mu_{\mathbf{D}}(\textit{DOUBLE}) = \{\textit{'0.1'}, \textit{'-0.165'}, \dots\} \quad (5.46)$$

$$\mu_{\mathbf{D}}(\textit{ENUMERATION}) = \{\textit{'Detect'}, \textit{'YES'}, \textit{'NO'}, \dots\} \quad (5.47)$$

5.4.3.10 Definition of Triple Statements for the Assembly Use Case

Lastly, the instantiated object types and data types must be set in relation to each other by defining triple statements. For the assembly process, there are more than 350 triple statements. Thus, only a small excerpt is shown in this section.

$$(\textit{Start Name 'Start'}) \in \beta(\mathbf{mt}_{\textit{FlowScene-AssemblyProcess}}) \quad (5.48)$$

$$\begin{aligned} (\textit{ObjectSpace-Assembly Process Name 'ObjectSpace - Assembly Process'}) \\ \in \beta(\mathbf{mt}_{\textit{FlowScene-AssemblyProcess}}) \end{aligned} \quad (5.49)$$

$$(\textit{Origin Name 'Origin'}) \in \beta(\mathbf{mt}_{\textit{FlowScene-AssemblyProcess}}) \quad (5.50)$$

$$(\textit{End Name 'End'}) \in \beta(\mathbf{mt}_{\textit{FlowScene-AssemblyProcess}}) \quad (5.51)$$

By showing that it is possible to formalize the introduced modeling method **ARWFMM** and the exemplary assembly process use case with the *FDMM* formalism, we were able to confirm the completeness and consistency of **ARWFMM** and are confident to state that the language could also be implemented on other modeling platforms.

5.5 Limitations Regarding Usability

In this chapter, we presented **ARWFMM**, a domain-specific visual modeling method that is capable of representing complex augmented reality workflows for diverse application scenarios and that can be executed using the open *WebXR* standard. The modeling method allows designers to specify three different types of visual models for defining (1) the **AR** environment, (2) the **AR** workflow, and (3) different statechanges within this workflow. Thus, the method emphasizes a high level of abstraction and separation of concerns. This abstraction bridges potentially missing knowledge about the technical implementation for **AR** environments and allows the user to focus on the content and functionality of **AR** applications. Technical feasibility was demonstrated by implementing the modeling language using the *ADOxx* platform and a prototypical web application to execute the models. A first evaluation has been conducted through (1) use case demonstration, (2) a feature comparison to previous approaches that indicated the high coverage of the defined requirements, and (3) formalizing the modeling language with the *FDMM* formalism, which shows the consistency of the modeling language.

The **2D** modeling approach presented in this chapter has some limitations with respect to usability due to modeling **3D** environments in **2D** modeling tools. As derived in the requirements in Chap. 4, when combining metamodeling and conceptual modeling with extended reality, it is necessary to access **3D** coordinate systems and to visualize **3D** models, i.e., SR1-SR2, and SR4-SR7—see Table 4.3.

For instance, accurately specifying the position of the legs in the aforementioned use case requires a thorough comprehension of three-dimensional space. It is nearly impossible to define the position and rotation vectors in **3D** space without visualizing them in **3D**. Figure 5.14 displays an *ADOxx* screenshot of the second last *Statechange* model from the furniture assembly workflow. The screenshot shows five *References* to *Augmentations* and each *Reference* defines the visibility, position, and rotation of the **3D** model represented by the *Augmentation*. However, it is not possible to determine these positions and rotations relative to the base coordinate system or relative coordinate systems—cf. Sect. 4.2—solely from this **2D** view. It would be easier to model these *Statechanges* in a three-dimensional modeling environment. Figure 5.15 shows an example of what such a *Statechange* visualization could look like in a three-dimensional environment. This approach would be more intuitive than the current **2D** modeling approach of *Statechanges* shown in Fig. 5.14.

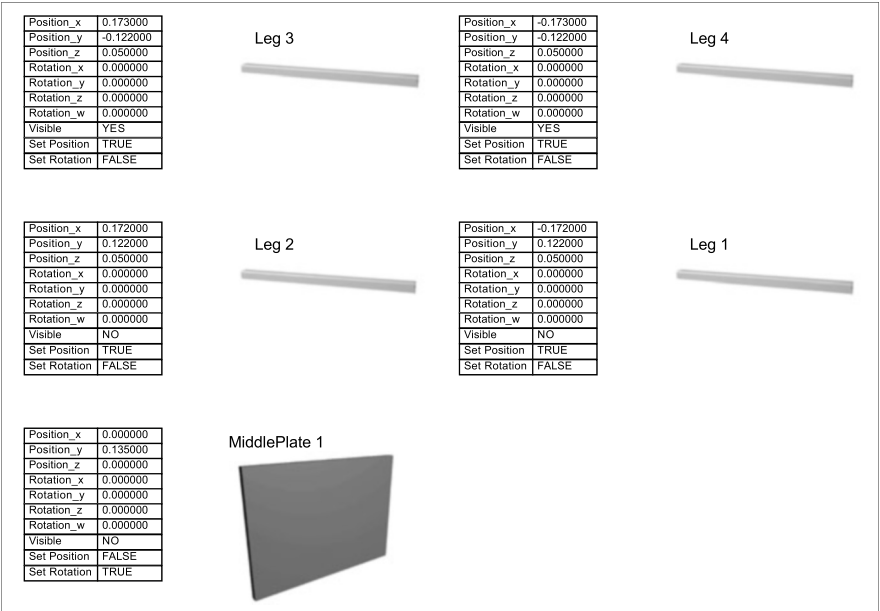
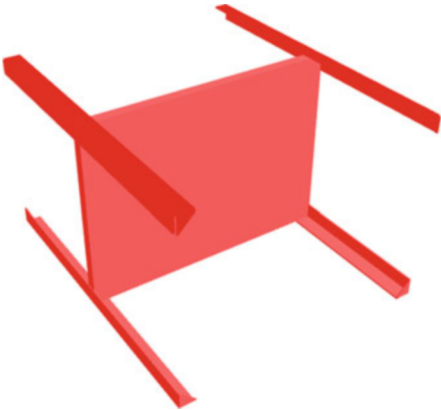


Fig. 5.14 Statechange model of the second last *Statechange* of the furniture assembly process. The statechanges defines the position and orientation of the four *Legs* and the *MiddlePlate*

Fig. 5.15 Visualization of what the second last *Statechange* would look like if it could be modeled in a three-dimensional environment



Reverting again to the different aspects of pairing conceptual modeling with virtual and augmented reality (see Sect. 2.3.6.1), it becomes clear that, for the proposed approach of **ARWFMM**, two aspects must be considered. Both of them are classified in the area of knowledge-based **VR/AR**, i.e., on the side of *flexibility aspects* (see Fig. 5.16). (1) There is the need of a *design-time* tool for visually modeling **AR** applications. This means that there is a need of a **3D**-enabled conceptual modeling tool, allowing to model on the basis of the proposed **ARWFMM**. (2) There is

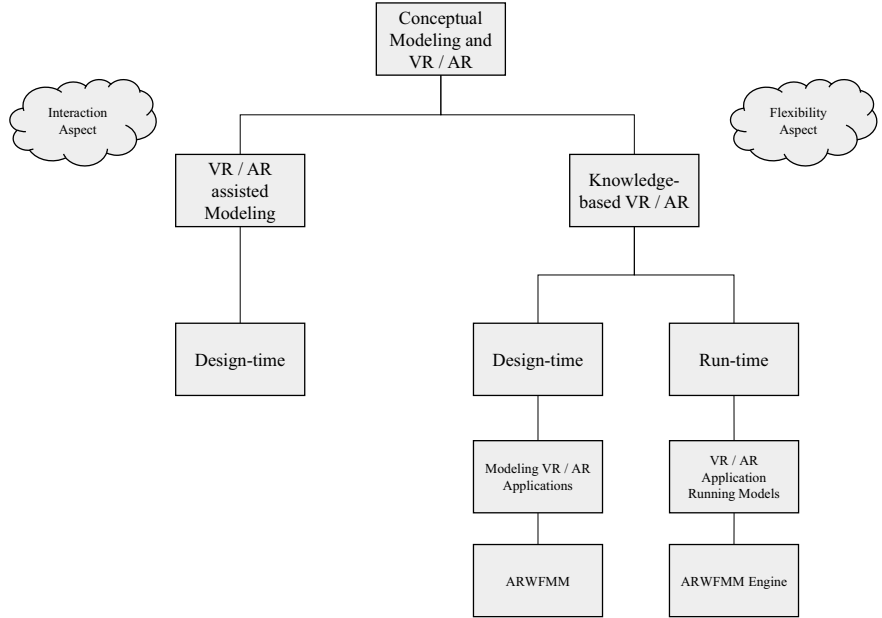


Fig. 5.16 Classification of the proposed components of **ARWFMM** and the **ARWFMM** Engine, based on the different aspects of pairing conceptual modeling with virtual and augmented reality

the *run-time* aspect, where **VR** or **AR** applications are taking models as input for creating **3D** experiences. This means that there is a need for an **AR** engine that is capable of interpreting models that are specified on the basis of the **ARWFMM** language schema. A prototypical implementation of this second part has already been shown in this chapter.

Taking into account the limitation of the proposed approach in modeling *Statechanges* without the third dimension, traditional two-dimensional modeling applications, such as the ADOxx modeling platform, are not appropriate for using **3D** modeling methods, such as the **ARWFMM**. Therefore, a new approach for **3D**-enabled conceptual modeling is needed, i.e., a **3D**-enabled metamodeling platform that incorporates the third dimension during visual modeling, thereby enabling **3D** modeling in three-dimensional space, cf. (Muff and Fill 2021a). Thus, the next chapter introduces a conceptual proposal for such a metamodeling platform.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



Chapter 6

M2AR: An Architecture for a 3D Enhanced Metamodeling Platform for Extended Reality



Some parts of this chapter have been published in a similar form as a research paper in: *Proceedings of the ER Demos and Posters 2021 co-located with 40th International Conference on Conceptual Modeling (ER 2021)* with the title: **Initial Concepts for Augmented and Virtual Reality-based Enterprise Modeling** (Muff and Fill 2021a).

As stated in the previous chapter's conclusion, 2D metamodeling platforms such as ADOxx (see Sect. 5.2.2) are not suitable for modeling three-dimensional AR applications, such as modeling with the newly introduced ARWFMM. However, not only for visual modeling of AR scenarios, such 2D metamodeling platforms come to their limits. Taking again into account the derived use cases for extended reality applications combined with conceptual modeling in Chap. 3, it becomes clear that the traditional 2D approach is not sufficient here either.

For example, the discussed use case for the strategic perspective (refer to Sect. 3.1.3) clearly shows that the lack of three-dimensional coordinates makes it impossible to model with a traditional metamodeling platform as a basis. The same is the case for the use case for the business process perspective (see Sect. 3.1.4). It is theoretically possible to base such a use case on 2D process models defined in a 2D modeling tool by adding additional 3D attributes as annotations to the modeling language. However, if information is needed that requires a relation to coordinates or orientation in the real world, the same problems as described at the end of Chap. 5 arise. This also applies to the third use case from the IT perspective (refer to Sect. 3.1.5).

Taking into account these limitations of 2D metamodeling platforms and the requirements for a metamodeling environment derived in Chap. 4, this chapter introduces the conceptual architecture for a 3D enhanced metamodeling platform that considers XR and is denoted as M2AR. The term M2AR is ambiguous. It can either be interpreted as *Model-To-AR*, which refers to a modeling environment for modeling AR applications, or for M^2 -AR, i.e., AR metamodeling, which covers the overall context of this work.

Karagiannis and Kühn (2002) defined a generic architecture to which every metamodeling platform should conform (see Sect. 5.2.2). However, there is a problem with this structure. Since all the modeling bases, i.e., the meta²-model, the metamodel base, the model base, and the mechanism base are enclosed by *persistence services and access services*, the architecture loses flexibility. Such a structure assumes that the whole metamodeling platform is one ecosystem and that all the viewer and builder applications, i.e., modeling and metamodeling clients, are dependent on data sources connected with the persistence service. The development of computer programs has undergone significant changes in the last 20 years. Therefore, we propose an updated, more flexible, and generic architecture for our metamodeling platform. This architecture still conforms to the architecture proposed by Karagiannis and Kühn (2002) in most parts.

6.1 Structure Base

There is a new component introduced, denoted as *structure base*. The structure base contains all the bases, i.e., the meta²-model, the metamodel base and the model base. What is new here is that the structure base does define the data structure of the contained bases. Thus, all the surrounding components can access this structure base. Since this structure is modular, the other components are not dependent on other components than the structure base anymore. For instance, a modeling client is no longer dependent on access and persistence services that connect it to a data source. Instead, it could rely independently on the structure base and file imports to enable modeling.

This is mainly useful if thinking of web-based applications being either independent or connected to a back-end server application, e.g., via a web API. Thus, the structure base builds the structural core of the metamodeling platform, and all other components are built around this structure base.

Another difference compared to the architecture proposed in Karagiannis and Kühn (2002) is the mechanism base. In this new proposal, the mechanism base is directly integrated into the meta²-model—see Sect. 6.1.1.8. Figure 6.1 shows a visualization of the newly proposed generic architecture for the M2AR metamodeling platform.

6.1.1 M2AR Meta²-Model: Meta-Layer

Inside the structure base, the meta²-model builds the core of the entire metamodeling platform. The meta²-model provides the basic concepts to create metamodels and mechanisms. Due to the missing 3D capabilities of traditional metamodeling platforms, we decided to introduce a new meta²-model, which will be described in the following. For the sake of comprehensibility, we separated the meta²-model

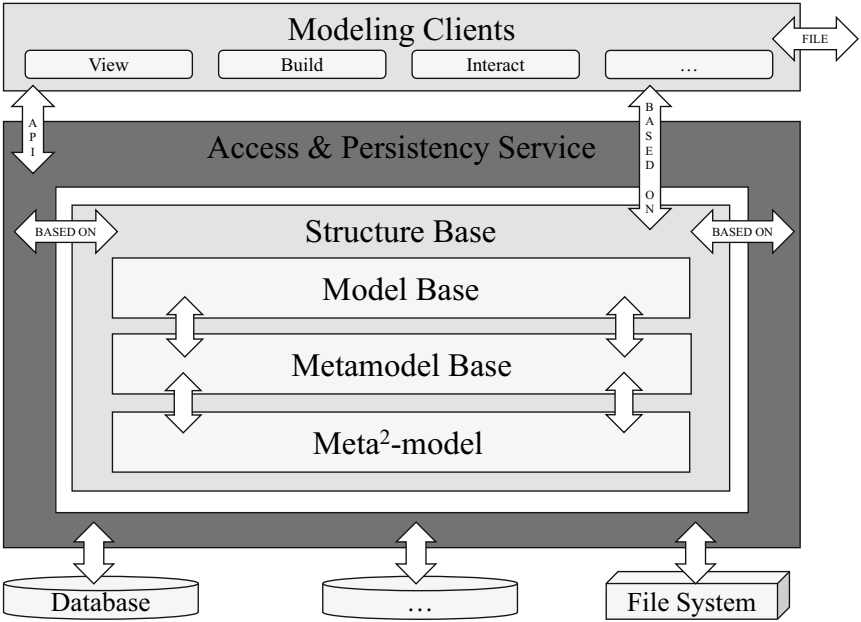


Fig. 6.1 Generic architecture of the M2AR metamodeling platform

into two parts. The meta-layer and the instance layer. Instances of the meta-layer correspond to the *metamodel base*, while instances of the instance-layer correspond to the *model base*—see Fig. 6.1.

For developing this meta²-model considering XR, we followed an exploratory and experimental research approach. In a first step, we analyzed the main concepts of existing meta²-models as described, for example, in Kern et al. (2011). This allowed us to identify the relevant concepts typically used in traditional 2D metamodeling. Subsequently, we derived the concepts necessary for XR in 3D space according to the requirements derived in Sect. 4.

The innovative aspect of the M2AR meta²-model is that it can be simultaneously used for two-dimensional and three-dimensional modeling languages. Unlike previous meta²-modeling approaches, it is natively based on 3D space (GSR1-GSR2). The meta²-model is composed of a meta-layer and an instance-layer. This is to show the relation between the definition of a modeling language and the instantiation of the specific objects when defining a model. Figure 6.2 only shows an *ER Diagram* of the meta-layer of the meta²-model. The instance-layer is visible in Fig. 6.3. The main concepts in the meta²-model on the meta-layer inherit the general properties from the superconcept *metaobject*.

6.1.1.1 Metaobject

Metaobject is the core concept of the meta²-model. It defines the common properties of all inheriting concepts. All concepts that can have a visual representation in a metamodel are inheriting from the *metaobject* concept. This visual representation is defined with a new domain-specific language called *VizRep*, that defines the 2D/3D representation and behavior of a visual object, under the consideration of well-known 3D data standards, e.g., the GLTF specification (**GSR7-GSR10**)—see Sect. 6.3.1.1.

This information is stored in the *geometry* attribute in *metaobject*. Furthermore, each visual object has 2D coordinates for positioning in a 2D modeling environment, as well as relative 3D coordinates (*relativeCoordinates3D*) for positioning objects in XR environments relative to the user position (**GSR2**). These positions may differ from the coordinates used for the 2D screen representation. Further, each *metaobject* may have absolute 3D coordinates (*absoluteCoordinates3D*) for the positioning of objects using real-world coordinates like Global Positioning System (GPS) coordinates or indoor positioning information (**GSR3**). Thus, **GSR1-GSR4** are covered. In addition to the different coordinates, there is also a property for the *rotation* of *metaobjects* (**GSR5**). Thus, all specific requirements for coordinate mappings are satisfied on a conceptual level—refer to Sect. 4.2. It should be noted that all of these properties on the meta²-layer apply to all instances of a *metaobject*. This means that if, for example, a *rotation* is specified at the *metaobject*, it will be applied by default to all instances of that *metaobject*. Therefore, such properties are only set at the meta²-layer if they are applied under all circumstances to all instances. Specific properties for instances are not considered on that level, but on instance-layer (see Sect. 6.1.2).

In addition to geometry, coordinates, and rotation, there are other properties that are common to all inheriting concepts of *metaobject*. There is a universally unique identifier (**UUID**) for identifying each *metaobject*. Furthermore, each *metaobject* has a *name* and a *description*.

Most concepts inherit from *metaobject*. These concepts are *class*, *role*, *scene_type*, *attribute*, *attribute_type*, *port*, *scene_group*, *user*, and *user_group*. In addition, a class can have the sub-concept *relationclass*, *decomposable_class*, and *aggregator_class*.

Classes are contained in one or multiple *scene_types*. A *scene_type* represents the closed model space, which is at the same time a closed 3D space. *Classes*, *ports* and *scene_types* have *attributes* that are further detailed with exactly one *attribute_type*. *Classes*, *ports*, *attributes* and *scene_types* can be set in relation to each other by *relationclasses*. Each *relationclass* has exactly two *roles* assigned. A *from_role* and a *to_role*. Furthermore, each *role* has at least one reference to a *class*, *scene_type*, *port*, or *attribute*, that defines, to what this role can connect. In addition, *classes* and *scene_types* can have *ports*. In the following, the different concepts are described in more detail.

6.1.1.2 Scene Type

A *scene_type* is the entry point for every metamodel. *Scene_types* can have *attributes* and contain *classes*, *relationclasses* and can have *ports*. Furthermore, a *scene_type* has a property defining whether it is a view with the “is-view” property. Views can restrict which visual concepts are available to show in this view. *Scenes* can be defined as subscene of other *scene_types* with the *is_sub_scene* relationship.

6.1.1.3 Class

A class is the most basic concept of the *metaobject* sub-concepts. This concept is identical to the known class concept described in Karagiannis and Kühn (2002). Considering known modeling languages such as ER diagrams (Chen 1976), the concepts “Entity”, “Attribute”, and “Relationship” would be typical instances of the M3 concept *class* in the new meta²-model. A class can specify the properties “is-reusable” and “is-abstract”. The class concept has three sub-concepts. *Relationclass*, *Aggregator Class*, and *Decomposable Class*.

6.1.1.3.1 Relationclass

Relationclass is the typical concept to connect multiple other concepts in a metamodel with a visual connector. Considering again the well known modeling language of ER diagrams, the drawn lines between entities, relationships, and attributes are typical examples of the *relationclass* concept. In contrast to most other meta²-models, we propose not to link *relationclasses* directly with the source and target concept of a relation, but to use an intermediate concept for more flexibility. Thus, a *relationclass* always specifies exactly one *from_role* and one *to_role* (refer to Sect. 6.1.1.4).

6.1.1.3.2 Decomposable Class

Decomposable Class is a concept for allowing the further decomposition of modeling concepts. This concept is similar to the GOPRR meta²-model of MetaEdit+ or the GME meta²-model of WebGME for abstracting complex metamodel concepts (see Sect. 5.2). A typical example of such a decomposition would be a BPMN task into a subtask, which can define a whole separate BPMN process. Thus, a BPMN class “subtask” would be a *decomposable_class* which defines in what *class* concepts or *scene_type* concepts the subtask can be decomposed.

6.1.1.3.3 Aggregator Class

Aggregator Class is the concept to describes spatial relationships between classes. For example, *is_inside*, *contains*, or *touches*. A typical use case where this is useful would be the pool class of the BPMN specification. A pool can contain multiple other classes. Thus, a BPMN pool class would be an *aggregator_class*. *Decomposable_class* and *aggregator_class* are both decorator-like extensions of *class* to define additional attributes that may be needed.

6.1.1.4 Role

The *role* concept is a very generic helper concept. *Roles* are used to connect other meta-concepts of the meta²-model. Thereby, a *role* can allow for references to *classes*, *relationclasses*, *scene_types*, and *ports*. All of these references specify a min and max cardinality for the specific reference, i.e., the source or the target end of a relation. This *role* concept is referenced by the above-mentioned *relationclass* concept and the *attribute_type* concepts (refer to Sect. 6.1.1.6).

6.1.1.5 Attribute

Scene_types, *classes*, and *ports* have *attributes*. The *has_attribute* relationships can specify the *sequence* and a *user interface (UI) component* value for potential connected graphical user interfaces during modeling. *Attributes* can have a *default value*, a *min* and *max* property to specify the minimal or maximum number of instances for this attribute, and a *facet* property to further constrain attribute values. *Attributes* can be constrained for editing by the *attribute_editing_constraint* relationship. Every *attribute* is of exactly one *attribute_type*.

6.1.1.6 Attribute Type

An *attribute_type* defines the data type of an *attribute*. It has a boolean value specifying if an *attribute_type* is *pre-defined* and a *RegExRule* (Regular Expression) to specify the rule for valid values for a specific *attribute_type*.

A special type of *attribute_type* is the *table_attribute*. *Attribute_types* can have other *attributes* of any *attribute_type* assigned to specify the columns of the *table_attribute* via the *has_table_attribute* relationship. There, the *sequence* number of each *attribute* (column of the table) are defined. One can assign any kind of *attribute* as column. Thus, potential tables in tables are possible.

Further, an *attribute_type* can reference a *role* with the *has_reference_role*, thus referencing very generically other *classes*, *relationclasses*, *scene_types*, and *ports*. This concept is similar to the *INTERREF* concept of the ADOxx meta²-model (refer to Sect. 5.2.2).

6.1.1.7 Port

A *port* can be seen as an interface on top of *scene_types* and *classes*. Instead of connecting *roles* from *relationclasses* and *attribute_types* directly to a *class* or *scene_type*, they can be connected to a *port* that is placed on a *scene_type* or *class*. This allows the connection options to be restricted more granularly.

6.1.1.8 Mechanisms and Algorithms

Unlike the design for the generic architecture of metamodeling platforms in Karagiannis and Kühn (2002), *mechanisms* and *algorithms* are not strictly separated from the meta²-model. In the following, the integration of *mechanisms* and *algorithms* into the newly proposed meta²-model is described (Bühlmann 2023).

An *algorithm* is a concept to run a predefined procedure upon user request on a conceptual model. *Algorithms* are stored in the *procedure* concept. *Procedures* are once defined and assigned to a *scene_type*. They are always specific for exactly one *scene_type*. A *procedure* has a property *definition* specifying the algorithms definition. An example of a modeling method-specific algorithm is the simulation of a business process model in BPMN 2.0 or the token game of a Petri Net. The definition of such algorithms is not restricted to a certain syntax. The interpretation of the value stored in the *definition* property is left to the metamodeling platform interpreting the meta²-model.

In contrast to *algorithms*, a *mechanism* is a concept used to run a process on instances of *scene_types* or *classes*. They are generically defined in the meta²-model in the *attribute_type* concept with the *mechanism_definition* property. Since *relationclasses* are subclasses of the *class* concept, *mechanisms* are also applicable for *relationclasses*. The idea behind *mechanisms* is that they should be run on every change of a concept attached to the respective *attribute_type* (*has_mechanism* relationship). This means that whenever an instance of an *attribute* is modified, or a new instance of an attached *class*, *relationclass*, or *scene_type* is created or deleted, all attached *mechanisms* should be executed. The execution of these *mechanisms* is not subject of the meta²-model itself, but of the implementing metamodeling platform.

6.1.1.9 User Group

Access rights are managed via the *user_group* and *user* concept. All *metaobjects* can be restricted via a *user_group* by assigning *read_access*, *modify_rights* and *create_rights*.

6.1.1.10 User

The management of users is based on the *user* concept. Each user has a login, consisting of a username and a password. Additionally, a user can have one or more *user_roles* assigned to them through the *has_user_role* relationship with a *user_group*.

6.1.1.11 Generic Constraint

Generic_constraint is a concept to add additional constraints or rules to a metamodel that cannot be enforced by the main concepts of the meta²-model. For example, to limit the number of allowed *start* elements in a BPMN model. *Generic_constraints* are independent of the *metaobject* concept and therefore have their own **UUID**, as well as a name and a value property. The content of this value property is not restricted, and the interpretation and execution of the defined rules is handled independently of the meta²-model.

6.1.2 M2AR Meta²-Model: Instance-Layer

The instance-layer of the M2AR meta²-model shows how instances of models (M1-level), conforming to metamodels (M2-level), are instantiated. On the meta-layer introduced before, modeling languages, i.e., metamodels are defined. For example, an instance of a class on the meta-layer is an instance in a metamodel (M2-level). Thereby, the construct *class* is the M3-level concept on meta²-model level. An instance of an M2-level concept, e.g., an instance of a class in a metamodel, is an instance in a model (M1-level).

In regard to Fig. 6.1, the meta-layer corresponds to the metamodel base. Instances of the specific concepts on this layer are then on the M1-level, i.e., the model level (model base). The main concepts in the meta²-model on the instance-layer inherit the general properties of the superconcept *instanceobject* (see Fig. 6.3). Each *instanceobject* has exactly one subconcept that is an instance of the according meta-layer concept, i.e., *class*, *relationclass*, *role*, *scene_type*, *attribute*, *attribute_type*, or *port*. Figure 6.3 shows an *ER Diagram* of the instance-layer of the newly proposed M2AR meta²-model.

6.1.2.1 Instanceobject

Every *instanceobject* has a **UUID** for identification and a *name*, as well as a *description*. Further, each *instanceobject* has **2D** coordinates for positioning in a **2D** modeling environment, as well as relative **3D** coordinates (*relativeCoordinates3D*) for positioning objects in **3D** environments relative to the user position (**GSR2**).

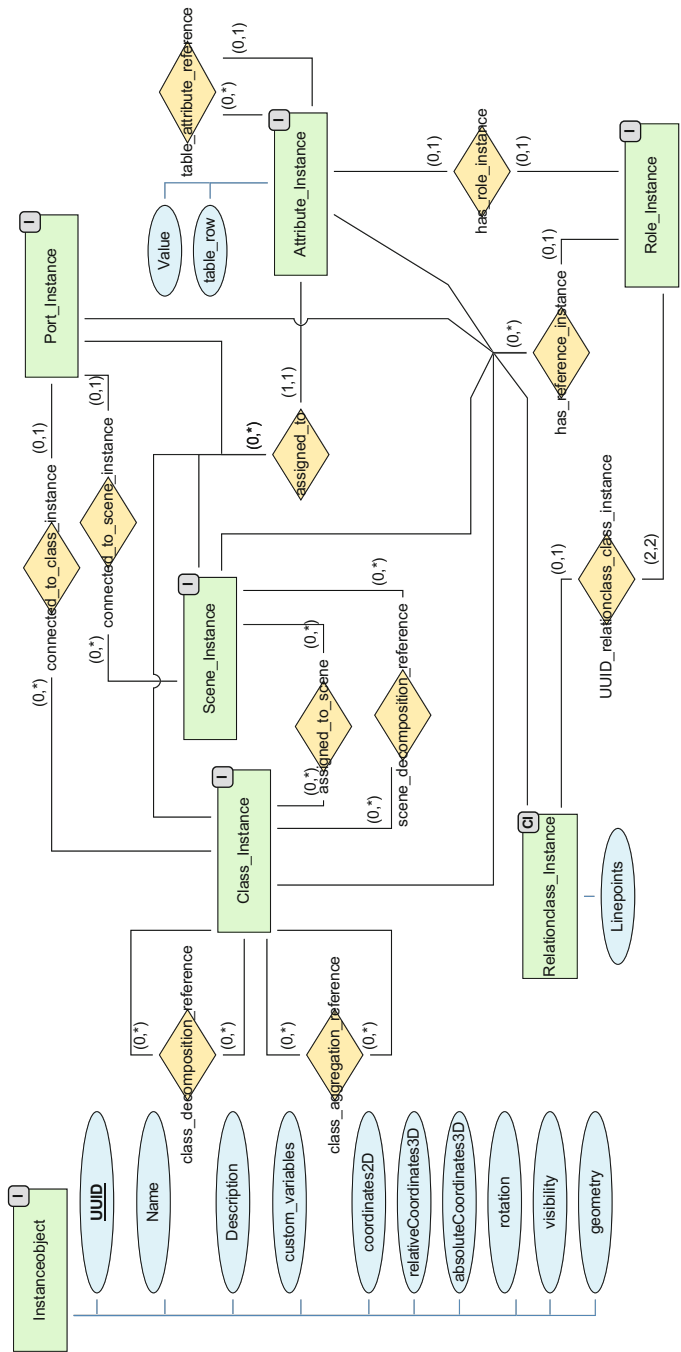


Fig. 6.3 ER diagram of the instance-layer of the newly proposed *M2AR* meta²-model. The (I) symbol signifies that the entity *IS-A instanceobject*. The (CI) symbol signifies that the entity *IS-A class_instance*

Again, these positions may differ from the coordinates used for the 2D screen representation. Further, each *instanceobject* can have absolute 3D coordinates (absoluteCoordinates3D) for the positioning of objects using absolute coordinate systems (GSR3). Thus, GSR1-GSR4 are covered as well on the instance-layer. In addition to the different coordinates, there is again a property for the *rotation* of *metaobjects* (GSR5), as well as a *visibility* property and a property for storing *custom_variables*. Further, *instanceobjects* hold again a geometry property for storing visualization information which do not hold for all instances (definition on meta-layer), but only for one specific instance. Again, all specific requirements for coordinate mappings are satisfied on a conceptual level—refer to Sect. 4.2. It should be noted that properties on the instance-layer, unlike properties on the meta-layer, are separately specified for every instance.

6.1.2.2 Scene Instance

The *scene_instance* is the entry point of every model instance. It defines the closed space of a model instance, i.e., a conceptual model. A *scene_instance* has a direct link to the meta-layer, i.e., to the *scene_type* concept of the meta²-model (see 6.1.1.2). A *scene_instance* can have connected *port_instances*, as well as assigned *attribute_instances* for specific attribute definition (*assigned_to* relation).

6.1.2.3 Class Instance

A *class_instance* is the most basic concept of the *instanceobject* sub-concepts. A *class_instance* can, but is not required to be assigned to different *scene_instances*. Furthermore, a *class_instance* can be decomposed into another *scene_instance* or other *class_instances* via *class_decomposition_reference* and *scene_decomposition_reference* relations. This corresponds to the instantiation of the meta-layer concept of *decomposable_class*. The same is true for aggregations via the *class_aggregation_reference*. In addition, the *class_instance* has a subclass *relationclass_instance* which relates to the meta-layer concept *relationclass*.

6.1.2.3.1 Relationclass Instance

Relationclass Instance has an additional property called *linepoints*. *Linepoints* specify the references to all the points of a visual *relationclass_instance*, which can be any *class_instance*, as long as this is specified on the meta-layer for the meta-model. Further, a *relationclass_instance* specifies two references to a *role_instance* (UUID_relationclass_class_instance) which correspond to the allowed *from_role* and *to_role* instances specified in the meta²-model.

6.1.2.4 Role Instance

The *role_instance* concept relates to the *role* concept on the meta-layer. *Role_instances* are used to connect other instances, such as *class_instances*, *relationclass_instances*, *attribute_instances*, *port_instances*, and *scene_instances*.

6.1.2.5 Attribute Instance

Attribute_instances correspond to the *attribute* concept of the meta²-model. They are used to attach specific attribute *values* to other instances, such as *scene_instances*, *class_instances*, and *port_instances*. An *attribute_instance* can be used inside a table instance via the *table_attribute_reference*. If an *attribute_instance* is used inside a table, the property *table_row* must be assigned to specify the row inside the table. Furthermore, an *attribute_instance* can reference a *role_instance* via the *has_role_instance* to reference other instances, such as *class_instances*, *relationclass_instances*, *attribute_instances*, *port_instances*, and *scene_instances*.

6.1.2.6 Port Instance

A *port_instance* relates to the *port* concept of the meta²-model. It is used as interface on *scene_instances* and *class_instances* via the *connected_to_class_instance* and *connected_to_scene_instance* relations. *Port_instances* can be connected via *role_instances* to other instances, i.e., *relationclass_instances* and *attribute_instances*.

6.2 Access and Persistency Service

Access Services and *Persistency Services* are core elements of every metamodeling platform. They manage the storage and access of all model and metamodel information (see Sect. 5.2.2).

These services provide clarity about the types of concrete storage, such as specific databases and file systems, and enable the distribution of components of models and metamodels (Karagiannis and Kühn 2002). In addition, access rights are also managed in these components.

In relation to this work, this means that there is a need for services that can handle the storage and the exchange of data to databases, file systems, or other information storage services. As visible in Fig. 6.1, we propose a module as *access & persistency service* that is based on the *structure base*, i.e., is strictly based on the predefined data structure according to the bases contained. Thus, a state-of-the-

art communication module is proposed to provide generic data access and database handling.

This service is used not only to grant access to data through the previously introduced user concept (see Sects. 6.1.1.10 and 6.1.1.9), but also to enforce additional rules that are not directly verifiable in the data structure. For example, if we again take the example of a BPMN process, we can define in the metamodel that it is allowed to connect a start event *class_instance* and a task *class_instance* via a subsequent *relationclass_instance*. We can also define that there is only one outgoing *relationclass* allowed from the start event *class_instance*. However, it is currently not possible to restrict that there is only one start *class_instance* allowed in the whole *scene_instance*. This could be checked in a modeling client during run-time via mechanisms or algorithms. However, this cannot ensure data consistency on the data layer, since it is not mandatory to use a modeling client. If such rules are to be enforced as data consistency rules on the data layer, they must be checked in the *access & persistency service*. Thus, the *generic_constraint* concept has been introduced into the meta²-model (see Sect. 6.1.1.11). There, generic rules to be checked in the *access & persistency service* can be generically defined.

Since the meta²-model does not provide a clear structure for the definition of rules, the *access & persistency service* can implement different interpretation engines for different rule notations. As discussed in Borcard (2022), there are different possibilities, e.g., the object constraint language (OCL) (Sunitha and Samuel 2018), or JSON-based rules.

6.3 M2AR Modeling Client

Based on the previously introduced *structure base*, and the *access & persistency service*, different *modeling clients* can be created. These modeling clients can have different purposes, e.g., viewing models, interacting with models, or creating and adapting model instances or metamodels. This includes the definition of modeling methods, including the definition of different language concepts, visual representations, mechanisms, and algorithms, as well as modeling itself.

6.3.1 Metamodeling Client

Before it is possible to model conceptual models with the help of a metamodeling platform, it is necessary to define metamodels, i.e., modeling methods. This includes all the aspects introduced in the meta²-model (see Sect. 6.1.1). It is possible that a metamodeler can do that by directly manipulating the data store of the whole environment, e.g., the database. Furthermore, a metamodeler can use the *access & persistency service* of a metamodeling environment, e.g., an *API* to create modeling methods. Both of these methods require a very deep understanding of the whole

structure and behavior of the entire ecosystem. Thus, these are not convenient ways to define modeling methods.

Taking into account that there is a need for a metamodeling client that allows the metamodeler to define modeling methods in an easy way (**GSR20**), a metamodeling client is needed. An example of such a metamodeling client is the *ADOxx Development Toolkit*,¹ which allows the definition of modeltypes, classes, relationclasses, attributes, as well as the visual representation and dynamic behavior of these concepts—see the ADOxx meta²-model in Fig. 5.3. Since we proposed a new meta²-model considering all the new requirements for 3D enabled metamodeling, there is a need for a new metamodeling client that conforms to the *structure base* and allows the interaction via a proposed *access & persistency service* to create modeling methods.

This includes the possibility to define all the required meta²-model concepts defined in Sect. 6.1.1, e.g., *classes*, *relationclasses*, or the definition of mechanisms and algorithms. One very specific requirement for this client application is the need to define visualization of 3D objects and their behavior on the meta-layer, i.e., the *geometry* property of *metaobjects* (**GSR6-GSR10**). Thus, we propose a new domain-specific language to define the visual representation of *metaobjects* in context of the introduced meta²-model in Sect. 6.3.1.1.

6.3.1.1 VizRep

The *VizRep* is a specialized domain-specific language (**DSL**) designed for the specific purpose of defining the visual representation of all model concepts on the meta-layer that are visually represented in a model. Thus, for each *metaobject* (see Sect. 6.1.1.1), the *geometry* property can be filled with a *VizRep* value that defines the visual representation of *instanceobjects* of that *metaobject*. The *VizRep* does not only define the static visual representation of an object, but also needs functions to access context information during modeling, e.g., values of *attribute_instances*, or information of the related *metaobject*.

The *VizRep* has a predefined set of methods that can be used to create custom visual representations. To avoid learning additional languages only for the definition of visual representations, the *VizRep* is based on the JavaScript² programming language. Thus, in a *VizRep* definition, all known concepts like loops, conditions, variables, etc. can be used. This also leads to some security problems, since this could be used as injection points for malicious JavaScript code. However, in this work, we will not go into more security-related details. The set of available language constructs is described in the following.

The *VizRep* is basically a normal asynchronous JavaScript function that is called during run-time in the modeling client used when instantiating an *instanceobject*.

¹ <https://www.adoxx.org/live/introduction-to-adoxx> last visited on: 01.03.2024.

² <https://tc39.es/ecma262/> last visited on: 01.03.2024.

It should be noted that the concepts introduced in this section do not constitute an implementation of the *VizRep*, but rather the conceptual structure of the language. The actual implementation and interpretation of the *VizRep* is left to a potential metamodeling or modeling client. The code box below shows the base structure of the *VizRep*.

VizRep base structure.

```
1 vizRep() { ... }
```

6.3.1.1.1 Dynamic Attribute Values

Dynamic Attribute Values are used to get the value of run-time *attribute_instances*. For example, if the visual representation of a *class_instance* should additionally visualize a label with the value of an *attribute*, this value must be retrieved via this concept. The keyword to obtain dynamic values is **dynval**. The *dynval* method takes two parameters as input. The **UUID** of the *attribute* (meta-layer), as well as the **UUID** of the *instanceobject* (instance-layer) the *attribute_instance* is attached to. The following code box shows the base structure of the *dynval* method.

Dynval base structure.

```
1 dynval(attribute_uuid : UUID, instanceobject_uuid : UUID)
```

6.3.1.1.2 Get Current Instance

Get Current Instance is a method for retrieving context information while modeling. It returns the **UUID** of the current instance with which the modeler is interacting in the visual modeling client. There are separate methods for different subclasses of *instanceobjects*. The code box below shows the different methods to get the instance context.

Available concepts for getting current instances.

```
1 get_current_scene_instance_uuid()  
2 get_current_class_instance_uuid()  
3 get_current_port_instance_uuid()
```

6.3.1.1.3 Variables

Variables can be defined inside the *VizRep* function like in a normal JavaScript function. Thereby, it is possible to store values as visible in the code box below.

Example for setting variables in VizRep.

```

1 vizRep() {
2   let example = 'hello world!';
3 }

```

This approach retrieves only the value once and stores it statically inside a variable. To define variables in a dynamic approach, it is possible to define them with a *set* and *get* approach. Thus, it is possible to dynamically access the variables at run-time. The code box below shows the method definition and an example of the dynamic approach.

Dynamic variables in VizRep.

```

1 setVariable(
2   name: string, value: string, instance_adaptable: boolean
3 ) { ... }
4
5 getVariableValue(name: string) { ... }

```

Combining the introduced concepts, it becomes possible to get dynamic values of run-time instances at the creation of a visual representation and store them into a variable. For example, if one creates a BPMN task *class_instance*, that has an *attribute_instance* of the attribute “name” (example [UUID](#) of the name *attribute* = “d6632c72-89fa-4210-9d01-18e911505608”), it is possible to get the instance value of this attribute instance dynamically, as visible in the code box below.

Example for the definition of dynamic variables.

```

1 setVariable(
2   'name',
3   dynval(
4     'd6632c72-89fa-4210-9d01-18e911505608',
5     get_current_class_instance_uuid()
6   ),
7   false
8 );

```

Thereby, the variable “name” can be used later in the *VizRep* code as follows:

Example for the use of dynamic variables.

```
1 vizRep() { getVariableValue('name') }
```

6.3.1.1.4 Graphic Cube

Graphic Cube is a concept to define a geometric visual cube as part of the visual representation of an instance. A cube always has a specific width, height, and depth. Furthermore, a cube can have a base-color, and a potential texture map in the form of an image that is mapped separately on each surface of the cube. Color and map are optional parameters to set. Thus, the method for defining a graphical cube in *VizRep* looks as follows:

Method for defining a cube in VizRep.

```
1 graphic_cube(  
2   width: number, height: number, depth: number,  
3   color?: string, map?: string  
4 ) { ... }
```

An example of the definition of a graphic cube in *VizRep* is visible below. Thereby, the variable “map” stores an image-string that is then defined as a texture map of the cube.

Example for defining a cube with a texture map in VizRep.

```
1 vizRep() {  
2   let map = 'data:image/png;base64,iVBORw0KG...';  
3   graphic_cube(0.1, 0.1, 0.02, 'grey', map);  
4 }
```

6.3.1.1.5 Graphic Plane

Graphic Plane is a very similar concept, except that this concept only defines one single geometric plane instead of eight connected planes. Thus, the plane method specifies only width and height, without depth. Further, color and map can be used like in the cube definition.

Method for defining a plane in VizRep.

```
1 graphic_plane(  
2     width: number, height: number, color?: string, map?: string  
3 ) { ... }
```

6.3.1.1.6 Graphic Sphere

Graphic Sphere is the third geometric concept that can be represented in the *VizRep*. A sphere is defined with a radius. Further, the degree of detail can be defined by setting the number of with segments and height segments of the sphere. Like the other concepts, a sphere can have a base-color, as well as a texture map. The method definition of a sphere in the *VizRep* is visible below.

Method for defining a sphere in VizRep.

```
1 graphic_sphere(  
2     radius: number, withSegments: number,  
3     heightSegments: number, color?: string, map?: string  
4 ) { ... }
```

Including the definition of 3D visualizations in the *VizRep* is covering **GSR8**. Note that multiple geometric objects can be combined in the same *VizRep* function. If this is the case, they will be combined into one graphical 3D object.

6.3.1.1.7 Standard Data Format

Standard Data Format support is one of the requirements derived in Chap. 4 (**GSR7**). Thus, it must be possible to use such well-known data formats as input for the graphical representation in the *VizRep*. For this purpose, the *VizRep* defines a method (*graphic_gltf*) to insert predefined GLTF files as a graphical 3D visualization. Since a GLTF file is essentially a string in a specific format, the method only requires a string as input. The following code box shows first the method definition of the *graphic_gltf* method, and second an example for inserting a GLTF as graphical object.

Method and example for inserting a GLTF in a VizRep definition.

```

1 graphic_gltf(objectString: string) { ... }
2
3 //example
4 vizRep() {
5   let gltf = 'your gltf sting';
6   graphic_gltf(gltf);
7 }

```

Since (1) [GLTF](#) files can contain dynamic animations, and (2) the generic definition of JavaScript functions allows for concepts such as loops and conditions, and (3) the definition of dynamic variables is possible in *VizRep*, the global specific requirement (**GSR10**) for the definition of dynamic behavior of model components is covered.

6.3.1.1.8 Graphic Text

Graphic Text is a graphic concept to create a text label with specific relative x, y and z position in relation to the objects origin (0, 0, 0). Furthermore, a graphic text has a size and height property, a color property, and four rotation properties (rx, ry, rz, rw), defining a quaternion for rotation. The “attribute” property defines the text content. Again, this is not only a static string, but can also be a dynamic attribute value as introduced above—see Sect. 6.3.1.1.3. This concept is used to visualize additional text labels, mostly in combination with other graphical visualizations. For example, a BPMN task could be visualized as a blue 3D cube with a 3D text label on top, visualizing the value of the *attribute_instance* of the assigned “name” attribute. The following code box shows the method definition and an example.

Method and example for defining text labels in VizRep.

```

1 graphic_text(
2   x_rel: number, y_rel: number, z_rel: number,
3   size: number, color: string, attribute: string,
4   rx?: number, ry?: number, rz?: number, rw?: number
5 ) { ... }
6
7 //example
8 graphic_text(
9   0, -0.1, 0, 0.2, 'gray', 'example text'
10 );

```

6.3.1.1.9 Graphic Line

Graphic Line is the concept to define the visual representation for *relationclasses*, i.e., mostly lines. It is to note that the *VizRep* does not define an instance of a line, i.e., a start and an end point, but only the style of the line including start and end connectors.

VizRep determines a line's representation based on several key attributes. The "color" attribute, defined as a string, specifies the line's visual hue using formats like hexadecimal codes or RGB values for precise coloration. The *line_width* attribute, a numerical value, dictates the thickness of the line, influencing its visual impact and clarity. For lines intended to be dashed, the *dashed* attribute, a boolean, indicates this style, switching between solid and dashed appearances. The characteristics of the dashed pattern can be further refined by adjusting *dash_scale*, *dash_size*, and *gap_size*. *Dash_scale*, which is a number, affects both the dash and the gap lengths, adjusting the overall scale of the dash pattern. *Dash_size* specifically determines the length of the individual dashes, allowing for customization of the dash pattern. Lastly, the parameter *gap_size* quantifies the space between the dashes, which plays a crucial role in the rhythm and spacing of the pattern. This completes the comprehensive set of parameters that define a line's representation in *VizRep*. The code box below shows the method definition of a graphic line.

Method for defining lines in *VizRep*.

```
1 rel_graphic_line(  
2     color: string, line_width: number, dashed: boolean,  
3     dash_scale: number, dash_size: number, gap_size: number  
4 ) { ... }
```

A *relationclass*, is not only visualized by a *graphic line*, but also by a start connector and an end connector and potential text labels. Thus, the *VizRep* also specifies methods for the lines from-connector (*rel_from_object*) and the lines to-connector (*rel_to_object*). These two methods again encapsulate a separate 3D geometry. Thus, it is possible to generically define complex 3D objects with the concepts introduced above as a from- or to-connector of a line. The code box below shows some examples of the definition of from- and to-connectors of a line.

Methods for defining from- and to-connectors of a line in VizRep.

```

1 rel_from_object(object)
2 rel_to_object(object)
3
4 //example
5 rel_from_object(
6     graphic_cube(0.006, 0.006, 0.006, 'black')
7 );

```

In addition, it is possible to add text labels to a line. There are three possibilities for defining a text label at the beginning, middle, or end of a line via the *rel_graphic_text_from*, *rel_graphic_text_middle*, and *rel_graphic_text_to* concepts. These three methods again encapsulate a separate 3D geometry that is intended to be a text label. However, it could also be another introduced concept. The code boxes below show the methods available for text labels on lines, as well as a complete example of a graphic line definition.

Methods for defining text labels of relations in VizRep.

```

1 rel_graphic_text_from(object) { ... }
2 rel_graphic_text_middle(object) { ... }
3 rel_graphic_text_to(object) { ... }

```

Example for defining text labels of relations in VizRep.

```

1 //example
2 function vizRep() {
3     let map = 'data:image/png; base64, iVB0 ... LSUVORK5CYII=';
4     rel_graphic_line('black', 0.002, false, 0, 0, 0);
5     rel_from_object(
6         graphic_cube(0.006, 0.006, 0.006, 'white')
7     );
8     rel_to_object(
9         graphic_plane(0.1, 0.1, 'grey', map)
10    );
11    rel_graphic_text_from(
12        graphic_text(-0.40, 0, 0, 0.1, 0.01, 'A')
13    );
14 }

```

6.3.1.1.10 Port

Port is the concept to define visual representations for *ports* of the meta²-model, i.e., graphical objects attached to *class_instances* or *scene_instances*. The definition of ports is exactly the same as for classes. Thus, all the concepts introduced above can be used. Whether a *class* or *scene_type* has ports, is not defined in the *VizRep*. This is defined by the metamodel of the modeling language.

6.3.1.1.11 Graphic Icon

Graphic Icon is the concept of defining a graphical icon to be used as preview of *classes* and *relationclasses* in modeling clients. Thus, every *VizRep* that should have an icon can define a variable “icon” that contains a base64 image string. The code box below shows an example for the definition of an icon.

Defining icons in VizRep.

```
1 let icon = 'data:image/svg+xml;base64,PHN2...';
```

6.3.1.2 VizRep Previewer

In addition to defining visual representations in a structured manner on the meta-layer, it is also crucial to provide users with visual cues on how the defined visual representation appears. Simply allowing textual definition of the visual representation is insufficient, as it can be challenging to envision the actual visualization of the defined *VizRep* function. Therefore, an integrated visual preview tool, as proposed in Nydegger (2022), is necessary within the metamodeling client. This previewer should be directly integrated into the metamodeling client and shows a graphical 3D rendering of the geometry according to the defined *VizRep* function.

6.3.2 Instance Modeling Client

The modeling activity is the central aspect of every metamodeling platform, in addition to defining metamodels. Instances of conceptual models, i.e. *scene_instances*, can be created on the basis of previously defined metamodels and the introduced meta²-model. In a very rudimentary modeling client, modeling can be done in a textual manner simply by writing down the data structure in a form that conforms to the *structure base* and the *persistency & access service* can interpret. This method, while direct, demands a high level of expertise from the modeler in understanding

the intricacies of the data structure, the metamodel, and the meta²-model. It is much easier if a visual modeling client is provided. Furthermore, as concluded in Chap. 5, it is almost impossible to imagine and estimate coordinates in 3D space without visualizing them.

To address this, we propose the development of an instance modeling client specifically designed for the M2AR meta²-model and its associated metamodels. This client would be grounded in a three-dimensional framework, allowing the visualization and precise modeling of traditional 2D, and 3D conceptual models. The client must adhere to the *structure base* and communicate through the *persistence & access service*, which includes user management and consistency checks. Furthermore, the client should be able to run predefined mechanisms and algorithms defined on the meta-layer. Lastly, the client should be able to operate independently of any communication service, such as through the import or export of text files.

Taking into account the requirements resulting from Chap. 4, an instance modeling client should additionally consider the concepts described in the following.

6.3.2.1 Basic 3D Environment Capabilities

The instance modeling client must support a 3D coordinate system. Thus, it must integrate 3D libraries which natively will facilitate user interactions within the 3D space, allowing for precise selection, manipulation and interaction with model elements in a 3D context and support features such as ray casting. This improves the modeling experience and accuracy and covers GSR11 and GSR17.

6.3.2.2 Detection and Tracking

As discussed in Chap. 5, the proposed ARWFMM requests for an AR engine for executing model instances. Since the proposal of the instance modeling client in this work is already based on a 3D environment, it is possible to integrate the tracking and detection requirements directly into the instance modeling client, thus covering GSR12-GSR16. This integration will enable the client to support the specific requirements of ARWFMM, allowing for the creation and manipulation of models in a 3D environment. The combination of ARWFMM and the instance modeling client enables smooth transitions and use of models at various stages of development and application. Of course, this is not only specific to ARWFMM, but also for other modeling languages.

6.3.2.3 Run-Time Sensor Data

By enabling the definition of generic mechanisms and algorithms in the meta-layer, the instance modeling client can accommodate real-time sensor data without the

need for additional functionality. Thus, **GSR18-GSR19** are covered. This feature is essential for models that require dynamic interaction with the real world, such as those used in **IoT** applications, where sensor data play a crucial role.

6.3.2.4 Interaction Capabilities

The instance modeling client must support both traditional **2D** interaction, as well as desktop-based and immersive **3D** interactions, including **XR** environments (**GSR21**). Thus, the instance modeling client should be based on a technology stack and software architecture that allows traditional **2D**, as well as desktop-based and immersive **3D** interaction.

6.3.2.5 Collaboration Capabilities

The instance modeling client must support functionalities for real-time collaboration between multiple users, as described in **GSR22**. This includes collaboration while modeling on **2D** desktops, and modeling or executing models in **3D** immersive environments, such as with a **HMD**. Therefore, the instance modeling client must support functionalities for the synchronization of model data, as well as real-world environment data, between different modeling clients to allow for real-time collaboration, e.g., as introduced in the use case in Sect. 3.1.3.

After considering the necessary components for a metamodeling platform with extended reality capabilities in Sects. 6.1, 6.2, and 6.3, the following section presents a conceptual architecture for a new metamodeling platform that incorporates the concepts introduced.

6.4 Proposed Conceptual Architecture for M2AR

This section proposes a detailed conceptual architecture for a metamodeling platform that enables **XR** functionalities, taking into account the aspects introduced in Sects. 6.1, 6.2, and 6.3.

Figure 6.4 shows the specific conceptual architecture proposed for *M2AR*, an **XR** enhanced metamodeling platform. The different modules will be explained in the following.

6.4.1 Global Shared Datastructure Module

The core of the metamodeling platform is the *Global Shared Datastructure* module, which corresponds to the *structure base* introduced in Sect. 6.1. All other modules

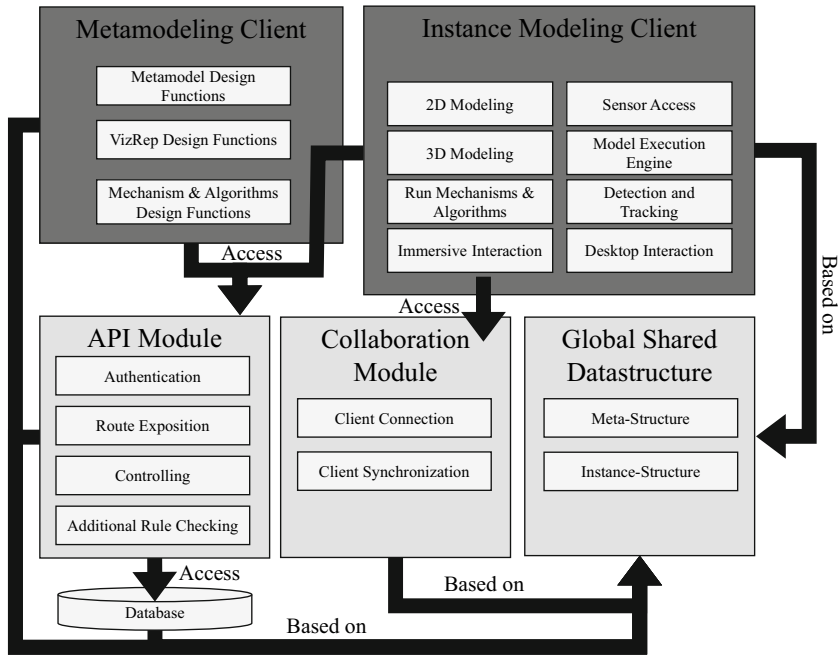


Fig. 6.4 Proposal of the specific conceptual architecture of the *M2AR* metamodeling platform

visible in Fig. 6.4 are based on this module, since it defines the data structure that is valid for the entire metamodeling platform.

6.4.2 Database

Very closely related to the *Global Shared Datastructure* module is the *Database*. The *Database* stores all the metamodel and model instance data according to the meta²-model defined in the *Global Shared Datastructure*.

6.4.3 API Module

On top of the *Database* there is an *API Module* that can access the *Database*. The *API Module* exposes different endpoints that are globally accessible by other modules. Further, the *API Module* handles user authentication, controls data consistency, and checks for additional rules defined in the metamodels.

6.4.4 *Metamodeling Client Module*

The *Metamodeling Client Module* is also based on the *Global Shared Datastructure* and uses the exposed endpoints of the *API Module* to access, modify, or create metamodels, including VizRep functions, mechanisms, and algorithms. In addition to defining the structure of metamodels through the *UI* forms, such as creating *scene_types*, *classes*, *relationclasses*, and *ports*, as well as establishing rules for the metamodels, it is necessary to integrate a graphical tool to define the *3D* visual representation of graphical objects through *VizRep*. Therefore, the client must combine a traditional *2D UI* with a *3D* framework for visualizing the previews of the *VizRep*.

6.4.5 *Instance Modeling Client Module*

The *Instance Modeling Client Module*, along with the *Metamodeling Client Module*, is based on the *Global Shared Datastructure*. It depends not only on the structure of instances, but also on the meta-structure, i.e., the meta-layer meta²-model. For modeling instances, previously defined metamodels are required as a basis. Such metamodels and instances are stored in the *Database* and can be retrieved to the client from the *API Module*. In addition, this flexible structure makes it is possible to import metamodels and model instances directly as text files, independent of a database or *API*.

The *UI* for the *Instance Modeling Client* needs a graphical interface to interact with traditional desktop-based *2D* equipment and state-of-the-art *XR* devices, such as *HMDs* or tablets. Furthermore, the modeling environment must be based on *3D* space, allowing for both *3D* modeling with a perspective camera view and *2D* modeling with an orthographic camera view. The perspective camera view mimics human-eye-depth perception with converging lines (*3D* perception), while the orthographic view maintains parallel lines and constant object sizes regardless of distance (*2D* perception). When modeling on a desktop, the user should be able to switch between views, depending on the work task to achieve. Thus, the *Instance Modeling Client* must be built with a technology stack that supports frameworks allowing for such a hybrid approach. Furthermore, the client must be able to access sensor data of the hosting device, (1) to allow interaction with *XR* technology, i.e., tracking—see Sect. 4.4—(2) for retrieving environmental context information—see Sect. 4.5—and (3) executing models via a built-in model execution engine—see Chap. 5.

6.4.6 Collaboration Module

The *Collaboration Module* is based on the *Global Shared Datastructure* module and is accessed from the *Instance Modeling Client*. The module should be capable of managing connections between various clients and tracking their manipulations of models or model states. Changes made to model instances must be propagated to all connected modeling clients as quickly as possible, preferably in near-real-time.

6.5 Implications for Implementation

After discussing the design of the new 3D enhanced metamodeling platform *M2AR*, we can now examine the implications of the different proposals for implementation.

Data Handling The purpose of a database is to store data. There are various database technologies available, such as SQL and noSQL databases, each with its own additional functionalities, such as triggers or functions. Therefore, there are multiple options to choose from. It is important to note that XR applications often rely on real-time adaptation of content to the environment. Thus, when selecting a database technology, the primary consideration should be optimizing performance for both writing and querying speed. The same applies to the proposed *API Module*, which can be used as an intermediate layer to communicate with the *Database*.

Common Technology Stack Furthermore, the different modules of the proposed architecture are basically independent and interchangeable. However, since all modules are based on the *Global Shared Datastructure*, it is important to find a common base technology stack for all modules to avoid unnecessary data conversion between modules.

Platform Independence Since the entire metamodeling platform must be accessible on a variety of different devices, it is important that the different modules exposed to different devices work platform-independent. Therefore, a technology stack must be found that is accessible to 2D desktop, phones, tablets, and HMDs. Furthermore, the technology must be capable of integrating 3D frameworks, as well as XR capabilities. Taking into account all of these requirements, there are limited options available. The most apparent solution to meet all of these requirements is to utilize web technology.

Interaction Since the metamodeling platform must be platform-independent and interaction must be possible with different devices, e.g. 2D desktops, 2D mobile devices, and HMDs, the technology used must support interaction with all these devices. Web technology does provide a good base for this, however, for 2D interaction with 3D environments and for XR interaction, additional 3D web frameworks are needed.

This chapter presented a first conceptual design for a 3D enhanced metamodeling platform for extended reality. The proposed platform consists of a *structure base*, a *persistency & access service*, and *modeling clients* for metamodels and instance models. In addition, a specific conceptual architecture for the platform was proposed, along with implementation implications. After considering all of these aspects, the next chapter discusses the first implementation of the M2AR 3D enhanced metamodeling platform that considers extended reality.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



Chapter 7

Prototypical Realization of the M2AR Metamodeling Platform



This chapter shows how the first prototype of the proposed *M2AR* metamodeling platform looks. The chapter is structured as follows. First, the technology stack used will be introduced (Sect. 7.1). Then, the different parts that have been implemented will be shown, including the database (Sect. 7.2), the *Global Shared Datastructure* (Sect. 7.3), the *API Server Module* (Sect. 7.4), as well as the *Instance Modeling Client* (Sect. 7.5).

7.1 Technology Stack

As described at the end of Chap. 6, the most obvious technology to use for a first implementation is web technology. Thus, the first implementation has been realized as a three-tier architecture system, encompassing a database server with *PostgreSQL*,¹ an *API* server running as *Node.js*² application and *express*,³ and a client web server running *Node.js* applications providing browser applications running *Aurelia*⁴ and the *JavaScript WebGL* visualization framework, *THREE.js*,⁵ in conjunction with the *WebXR device API* (Jones et al. 2023). For all the *Node.js* applications, we used *TypeScript*,⁶ a programming language that is strongly typed and builds on *JavaScript*, providing improved tooling at any scale. Figure 7.1 presents an overview of the architecture of the prototypical implementation of the new *M2AR* metamodeling platform, including the technologies used for the

¹ <https://www.postgresql.org/docs/> last visited on: 01.03.2024.

² <https://github.com/nodejs/node> last visited on: 01.03.2024.

³ <https://github.com/expressjs/express> last visited on: 01.03.2024.

⁴ <https://github.com/aurelia/aurelia> last visited on: 01.03.2024.

⁵ <https://github.com/mrdoob/three.js> last visited on: 01.03.2024.

⁶ <https://github.com/microsoft/TypeScript> last visited on: 01.03.2024.

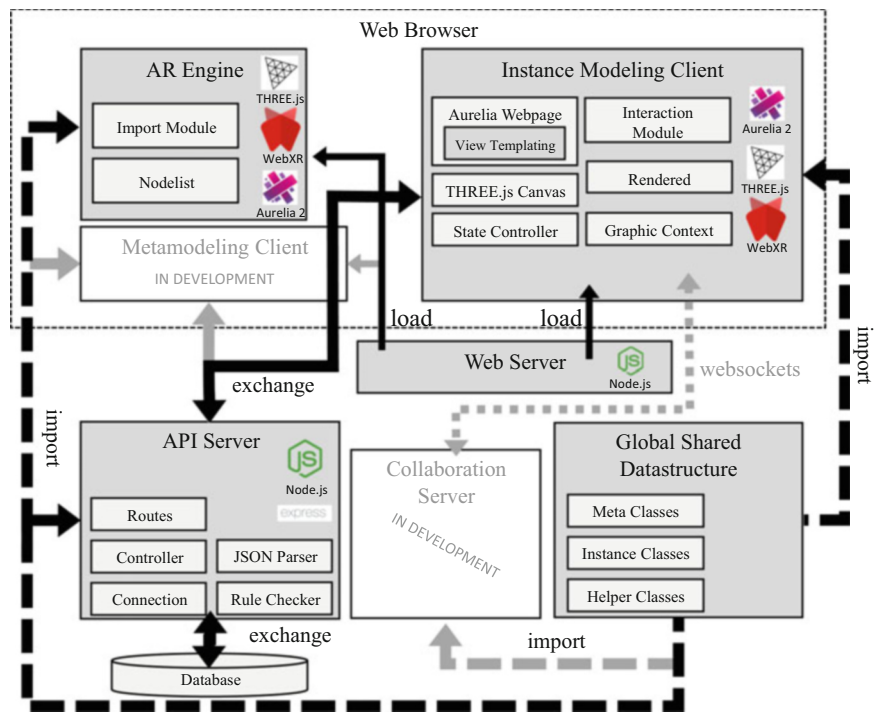


Fig. 7.1 Overview of the architecture of the new *M2AR* metamodeling platform implementation, including the technologies used for the different modules and the most important components of the modules

different modules and the most important components of the modules. The *M2AR Metamodeling Client* and the *Collaboration Server* module have not yet been fully implemented.

7.2 Database

PostgreSQL is an open-source relational database system that is reliable, robust, and performs well in handling complex data workloads. It is highly regarded for its strong compliance with structured query language (SQL) standards and its ability to handle various data types, including JSON. PostgreSQL’s structured database schema is crucial for preserving data integrity and accessibility within the metamodeling platform.

Table 7.1 Table names of the 52 tables implemented in the PostgreSQL database

Table names	
aggregator_class	has_write_right
assigned_to_scene	instance_object
attribute	is_sub_scene
attribute_instance	is_subclass_of
attribute_propagating_relationclass	metaobject
attribute_type	port
class	port_has_attributes
class_aggregation_reference	port_instance
class_decomposition_reference	propagation_attribute
class_has_attributes	relationclass
class_instance	relationclass_instance
contains_aggreg_classes	role
contains_aggreg_relationclasses	role_class_reference
contains_classes	role_instance
decomposable_class	role_port_reference
decomposable_into_aggregator_classes	role_relationclass_reference
decomposable_into_classes	role_scene_reference
decomposable_into_scenes	scene_decomposition_reference
file	scene_group
generic_constraint	scene_has_attributes
has_delete_right	scene_instance
has_read_right	scene_type
has_reference_role	selected_propagation_attributes
has_right	user_group
has_table_attribute	users
has_user_user_group	t_history

Thus, the *M2AR* meta²-model, including meta-layer and the instance-layer introduced in Sects. 6.1.1 and 6.1.2 have been implemented in a PostgreSQL database. The current implementation uses PostgreSQL version 15 which was released on October 2022. The database is divided into two schemas. The *logging* schema, which logs every transaction from the metamodeling platform, and the *public* schema, which holds all the data on the metamodells and model instances. At the time of writing, the public schema had a total of 51 tables. The *logging* schema contains only one table *t_history*. Table 7.1 shows the table names of all 52 tables implemented in the PostgreSQL database.

The database not only stores the schema and data itself. It also stores functions for functionalities that cannot be done using data constraints. There are the three functions *change_trigger()*, *delete_instance_parent()* and

delete_metaobject_by_uuid(uuid, uuid). *Change_trigger()* is a trigger function that logs changes such as inserts, updates, or deletes to the *t_history* table in the logging schema. *Delete_instance_parent()* and *delete_metaobject_by_uuid(uuid, uuid)* are trigger functions to ensure data consistency when deleting instances or meta-concepts which would lead to parent *metaobjects* or *instanceobjects* without children, which is not possible according to the meta²-model.

7.3 Global Shared Datastructure Module

The *Global Shared Datastructure* module is implemented as TypeScript class definitions. The module is not a running program, but only the definition of classes and implements the data structure according to the meta²-model. It is separated into two parts. The meta-layer and the instance-layer.

The meta-layer contains 20 TypeScript classes. *Metaobject* is the entry-point class, corresponding to the concept defined in the meta²-model. *Class*, *AttributeType*, *Port*, *Procedure*, *Role*, *SceneType*, *File*, *Usergroup*, and *User* are subclasses of *Metaobject*. *Relationclass* is a subclass of *Class*. All these classes are directly derived from the meta²-model. *ColumnStructure* is a helper class for table attributes and *ClassReference*, *RelationclassReference*, *SceneTypeReference*, and *PortReference* are classes for defining the cardinalities of the *Role* concept. Furthermore, there are the classes *Point2D*, *Point3D*, *Quaternion*, and *Rule* which are defining the structure of other independent meta²-concepts.

The instance-layer contains eight TypeScript classes: *AttributeInstance*, *ClassInstance*, *ObjectInstance*, *PortInstance*, *RelationclassInstance*, *RoleInstance*, *SceneInstance*, and *InstanceRowStructure*.

ObjectInstance and *InstanceRowStructure* are top-level classes. They do not inherit from any other class. *ObjectInstance* is corresponding to its instance-layer concept in the meta²-model. *InstanceRowStructure* is a concept that is needed to implement the table attributes at the instance level. *AttributeInstance*, *ClassInstance*, *PortInstance*, *RoleInstance*, and *SceneInstance* are inheriting from *ObjectInstance* according to the instance-layer concepts.

The entire module is distributed as a *Node.js* module, so that it can be easily imported into other modules depending on the *Global Shared Datastructure* module. Figure 7.2 shows a simplified TypeScript class structure of the *Global Shared Datastructure* module.

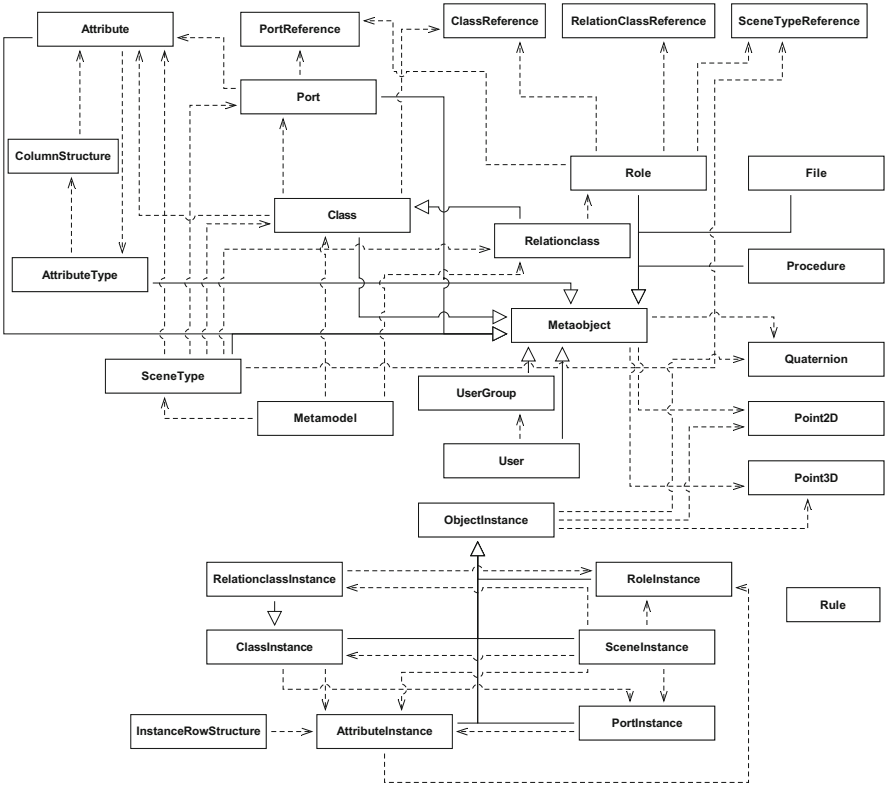


Fig. 7.2 Simplified class structure of the *Global Shared Datastructure* module

7.4 API Server Module

The [API](#) server module is implemented as *Node.js* application using *express*. The module exposes in total 151 REST (RESTful) [API](#) endpoints. 97 on the meta-level and 54 on the instance-level. This includes endpoints for POST, GET, PATCH, and DELETE functionalities. In addition, the module defines controller classes and connection classes for writing to, and retrieving data from the database.

The server module also implements middleware services for checking additional rules (see Borcard [2022](#) for more details) and for parsing the data into a JSON format that fits into the *Global Shared Datastructure*—see Sect. 7.3. Figure 7.3 shows a simplified visualization of the JSON structure of a metamodel. This schema is generated by the [API](#) server module and also takes this structure as input.

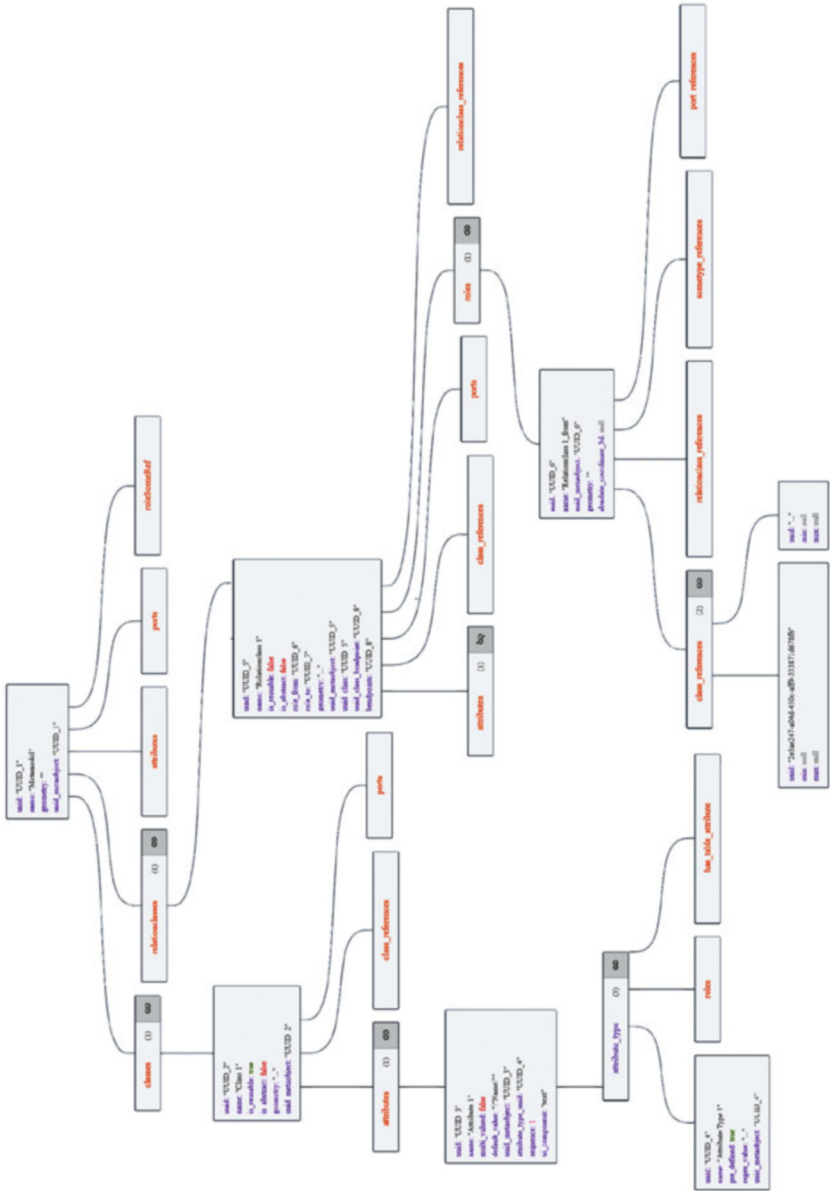


Fig. 7.3 Simplified visualization of the JSON structure of a metamodel

7.5 Instance Modeling Client: M2AR Modeler

The *Instance Modeling Client*, denoted as the *M2AR Modeler*, is the part with the most challenging module architecture. As described in the technology stack, the *M2AR Modeler* is a *Node.js* client application that exposes a client website for modeling. The UI of the *M2AR Modeler* is built with the *Aurelia 2* framework and the *THREE.js* 3D framework.

Aurelia 2 is a modern JavaScript/TypeScript framework that enables developers to create powerful and efficient applications. It adopts a convention-over-configuration approach and is highly modular and extensible, supporting a range of plugins and customizations. The framework emphasizes clean and testable code, taking advantage of modern JavaScript/TypeScript features and patterns. With *Aurelia 2*, developers can define modular components (templating).

The purpose of the *M2AR Modeler* is to create visual conceptual models based on predefined metamodels. Users can interact with the client to create and manipulate visual conceptual models in traditional 2D and in 3D. As proposed in Chap. 6, the *M2AR Modeler* is based on the *Global Shared Datastructure* module. Thus, the client initializes the data structure in a global context and creates and stores metamodels and instance models locally in that data structure.

The visualization of these instances in the *M2AR Modeler* is independent of the instances of the *Global Shared Datastructure*. Thus, it should be noted that the *M2AR Modeler* always instantiates two instances for each new instance of the *Global Shared Datastructure*. (1) The instance of the data structure itself, and (2) a graphical *THREE.js* instance of the 3D representation according to the *VizRep* definition defined in the metamodel—cf. Sect. 7.5.4.

7.5.1 Visual Main Components of the Client

As most traditional modeling clients, the *M2AR Modeler* is composed as a single page with four main components. (1) A top menu bar, (2) a left navigation bar, (3) a middle body, and (4) a right navigation bar. Figure 7.4 shows the modular structure of the *M2AR Modeler*.

Upon initialization of the page, all the standard components are composed. In the following, the different components within the four main components are explained. Figure 7.5 shows these different components. The screenshot visualizes a scenario where an instance of a BPMN model is created on the modeling canvas in the 2D view mode.

The **Top Menu Bar** is made up of multiple buttons and menu entries for navigation and modeling actions, e.g., deleting instances, saving models, or loading external files.

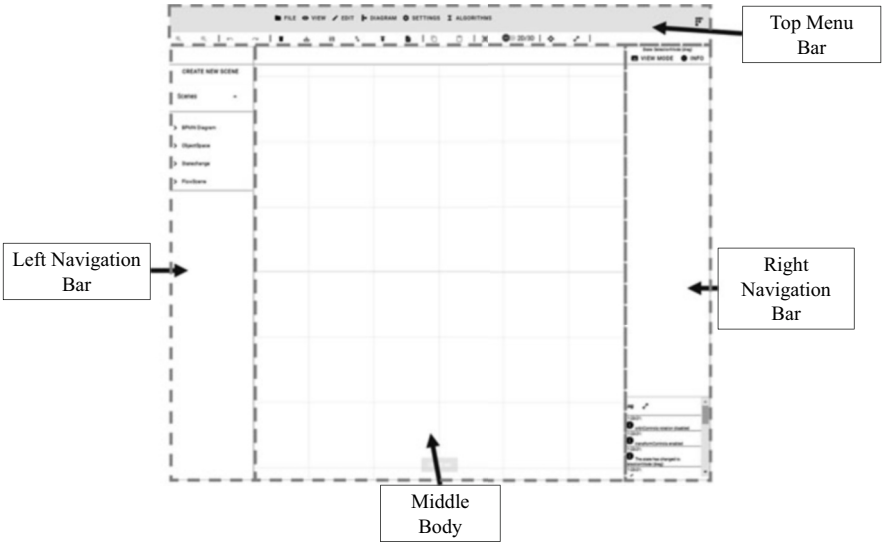


Fig. 7.4 Modular structure of the M2AR Modeler

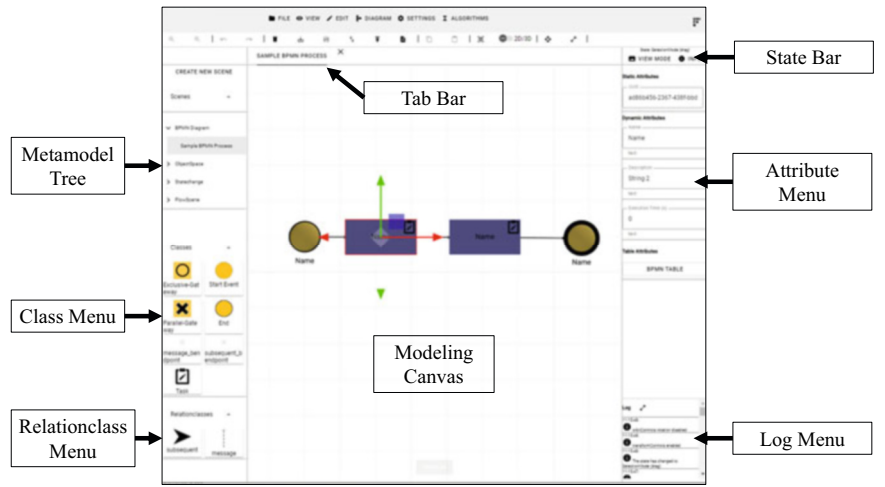


Fig. 7.5 Screenshot of the *M2AR Modeler* showing the different components of the client. The screenshot shows a scenario where an instance of a BPMN model is created on the modeling canvas in the 2D view mode

The **Left Navigation Bar** contains all the information about the metamodels and model instances available, as well as the specific *classes* and *relationclasses* available for modeling if a *SceneInstance* is opened. Thus, the *Left Navigation Bar* contains a section for the *Metamodel Tree*, a *Class Menu* and a *Relationclass Menu*. Both the *Class Menu* and the *Relationclass Menu* contain selectable icons for each *class* and *relationclass* defined in the selected metamodel.

The **Right Navigation Bar** contains dynamic components that depend on the modeled instances and the actions of the users. On top of the component, there is a *State Bar* that shows the current active modeling state; see Sect. 7.5.3. The *Attribute Menu* is below the *State Bar*. This menu is dynamically showing the *attribute_instances* of the instance selected in the *Modeling Canvas*. The *Log Menu* is on the bottom right. This menu shows log entries on the modeling activities to allow the user to track changes to the models.

The **Middle Body** contains the environment for modeling, i.e., the *Modeling Canvas*. It is possible to open multiple model instances and switch between the open models using the *Tab Bar*. The distinguishing feature of this modeling client, as opposed to traditional ones, is that the *Modeling Canvas* is not limited to a 2D surface, but instead is a complete 3D environment that spans a 3D coordinate system based on the *THREE.js* web framework. The standard modeling environment uses an orthographic camera environment, which means that the user cannot see that there is a 3D environment, but all model instances are simply placed in the 3D environment with a fixed position on the z-axis. For modeling languages that require 3D modeling, such as *Statechange* model instances introduced in the *ARWFMM* modeling language (cf. Chap. 5), the *Modeling Canvas* can be switched to a 3D view with a perspective camera and three-degree-of-freedom mouse interaction capabilities. Figure 7.6 shows a screenshot of the new 3D modeling mode allowing spatial modeling on a 2D screen. Another advantage of the technology stack used is that it is natively compatible with XR devices. Therefore, it is also possible to view the created models directly in VR or AR by using smartphones, tablets, or HMDs. Additionally, it would also be possible to allow model creation and manipulation in XR. However, the interaction features with the controller or the hand tracking (see Sect. 4.6) are not yet implemented.

7.5.2 Client Communication

The web client retrieves all available metamodels and model instances through the *API Server Module* and visualizes them in a tree view in the left navigation bar. Additionally, the client includes functionalities for sending newly created model instances to the database or updating existing models in the database via the *API*. The *M2AR Modeler* can also work independently of the database. It is possible to load metamodels as JSON files into the modeling client to model instances according to the imported metamodel. Model instances can be imported or exported as JSON files to and from the host device file system, if desired.

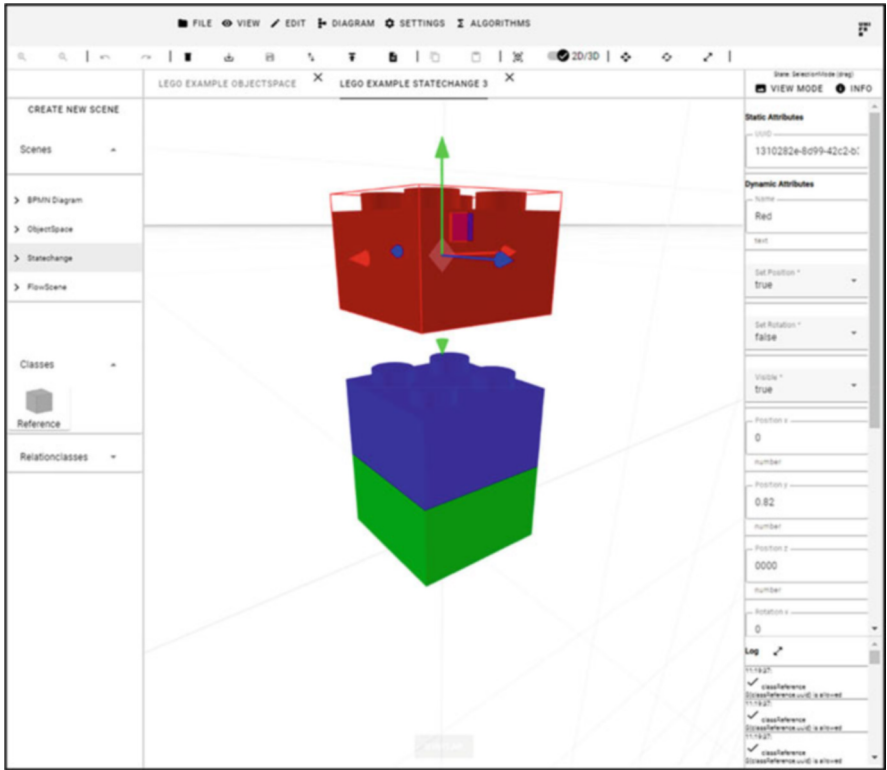


Fig. 7.6 Screenshot of the new 3D modeling mode in the *M2AR Modeler* allowing for spatial modeling on a 2D screen

The *M2AR Modeler* must also communicate with the *Collaboration Server* proposed in Sect. 6.3.2.5. However, this module is currently still in development.

7.5.3 Modeling States

The interaction of the *M2AR Modeler* is implemented with different modeling states. Depending on these modeling states, interactions with the client result in different actions. After analyzing different traditional modeling clients and comparing their modeling behavior, four main states have been considered for the implementation of the *M2AR Modeler*. These states are (1) *SelectionMode*, (2) *ViewMode*, (3) *DrawingMode*, and (4) *DrawingModeRelationClass*.

In the ***ViewMode***, no instance is selected. The user can zoom in or out and move the camera around. In the ***SelectionMode*** an instance of the model is selected. The user can still zoom in or out and can further manipulate the visualization of the

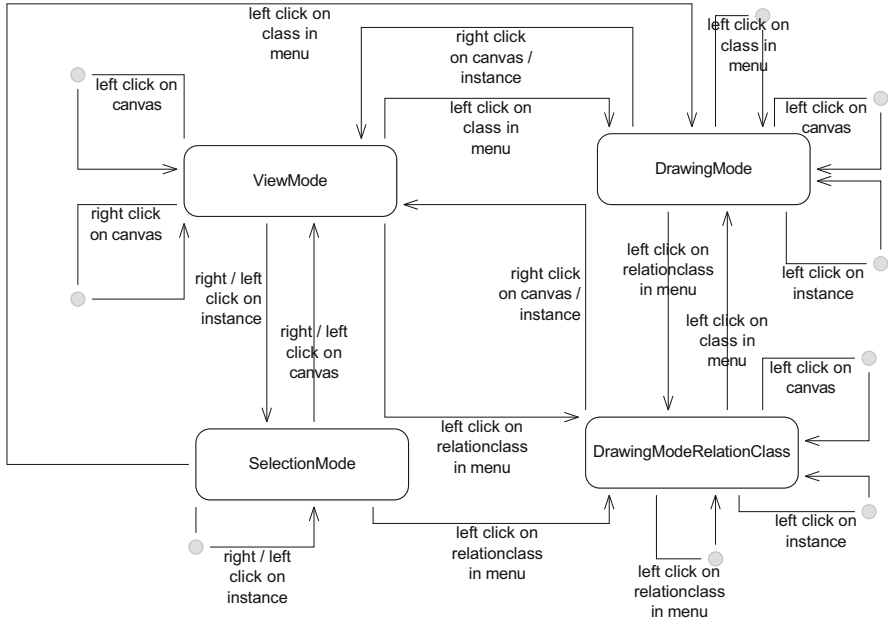


Fig. 7.7 State machine diagram of the different states implemented in the *M2AR Modeler* and the according transitions between the states

selected instance. This means that it is possible to translate, scale, or rotate the selected instance visualization. If a class from the *Left Navigation Bar* is selected, the client is in the **DrawingMode**. This allows the user to create new *class_instances* on the *Modeling Canvas*. Zooming in and out is also possible in this state. If a relationclass from the *Left Navigation Bar* is selected, the client is in the **DrawingModeRelationClass**. This allows for the creation of *relationclass_instances* by selecting two instances in the *Modeling Canvas*. When clicking on a free area after selecting the first instance, bendpoints are added to the *relationclass_instance*.

Figure 7.7 shows a state machine diagram of the different states implemented in the *M2AR Modeler* and the transitions between the states. It must be noted that the diagram illustrates numerous potential transitions between the states, which are not elaborated on in this section.

7.5.4 VizRep Implementation

An essential aspect of the *M2AR Modeler* is the implementation of the *VizRep*, which was conceptually introduced in Sect. 6.3.1.1. To implement the *VizRep*, a

new singleton class called *GraphicContext* (gc) was created. This class contains all the methods necessary to create *THREE.js* 3D objects. *THREE.js* is based on a 3D base coordinate system, as introduced in Sect. 4.2. Thus, each 3D object again spans its own 3D coordinate system that is placed in a parent coordinate system.

In the following, we present two examples of *VizRep* definitions and the corresponding 3D visualizations in the *M2AR Modeler*. Since the *VizRep* language is very generic, the *GraphicContext* class must be very flexible. During the execution of a *VizRep* function, all specific 3D objects are added to an array, and only at the end, when the object is drawn to the canvas, are they merged into a single 3D object.

VizRep Example 1: Example of the VizRep definition of a BPMN Task class

```

1  async function vizRep(gc) {
2      let gltf = '{ \"asset\": { \"version\": \"2.0\", \"gen...';
3      await gc.setVariable(
4          'name',
5          await gc.dynval(
6              'd6632c72-89fa-4210-9d01-18e911505608',
7              await gc.get_current_class_instance_uuid()
8          ),
9          false
10     );
11     await gc.setVariable('x_rel', 0, true);
12     await gc.setVariable('y_rel', 0, true);
13     await gc.setVariable('z_rel', 0.143, true);
14     await gc.graphic_gltf(gltf);
15     await gc.graphic_text(
16         await gc.getVariableValue('x_rel'),
17         await gc.getVariableValue('y_rel'),
18         await gc.getVariableValue('z_rel'),
19         0.2,
20         'black',
21         await gc.getVariableValue('name'),
22         'x_rel',
23         'y_rel',
24         'z_rel'
25     );
26     let icon = 'data:image/png;base64,iVBORw0KGg...'
27 }

```

The first example (see the code box “VizRep Example 1”) shows the *VizRep* definition of the BPMN class “Task”. The *VizRep* defines first a “gltf” variable with the JSON representation of a blue box with a black icon in the top right corner (code

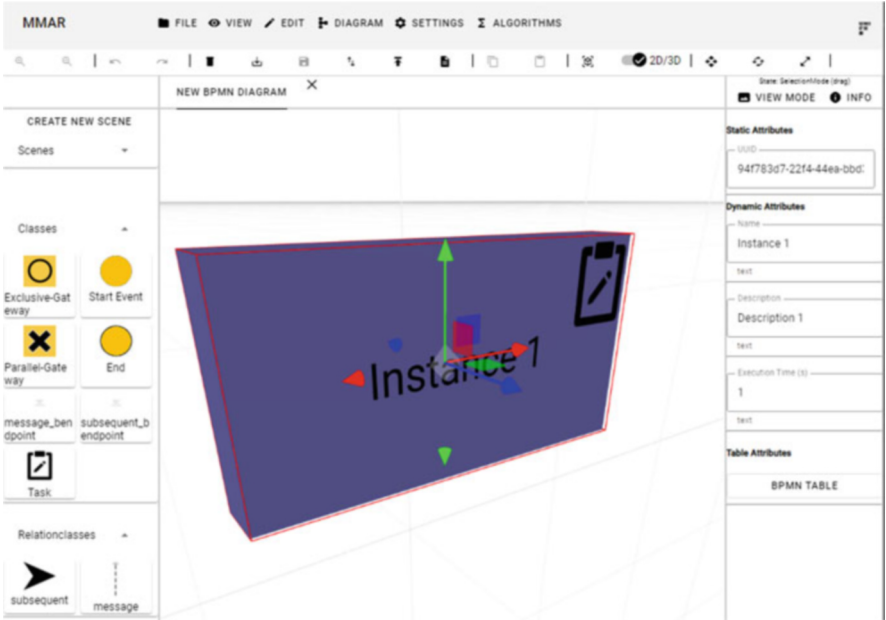


Fig. 7.8 Visualized 3D instance of a BPMN “Task” class_instance according to the VizRep definition in *VizRep Example 1*. As visible, the text label is instantiated according to the value of the “name” attribute_instance visible in the Attribute Menu

line 2). The GLTF box is two units large and one unit high, whereby one unit is equal to one meter. It must be noted that this 3D object must be created in advance in a 3D model editor, e.g., the THREE.js editor.⁷ Alternatively, the standard concepts for creating cubes, spheres, lines, or planes may be used—see Sect. 6.3.1.1. In code lines three to ten, a dynamic variable is defined that takes the attribute_instance value of the “name” attribute as value. Furthermore, three variables x_{rel} , y_{rel} , and z_{rel} are set (code line 11–13). Code line 14 creates a 3D instance of the “glTF” variable from line two. In addition, a 3D text label is instantiated, taking the three positioning variables x_{rel} , y_{rel} , and z_{rel} as input for the relative position of the text label, and the dynamic variable “name” from line five as a value to display as text label. Line 26 of the code defines the variable for the icon of the image that the client uses in the *Class Menu*.

As visible in Fig. 7.8, the visualized 3D instance of a BPMN “Task” class_instance according to the *VizRep* definition in *VizRep Example 1* shows a

⁷ <https://threejs.org/editor/> last visited on: 01.03.2024.

blue 3D cube and the text label is instantiated according to the value of the “name” *attribute_instance* visible in the *Attribute Menu*. The three arrows on the 3D object do not belong to the object itself, but are only visible if an instance is selected in the *SelectionMode* and allow for the transformation of the object’s position.

Looking at another example, it becomes apparent that the *VizRep* language can be used very generically and always works the same for all the 3D concepts. The code box “VizRep Example 2” shows an example of the *VizRep* of the “triggers” *relationclass* from the *ARWFMM*—see Chap. 5. The “map” variable on code line two defines a 2D image in *base64* format. The “icon” variable on line three takes this “map” variable as input for the *Relationclass Menu* icon of the *M2AR Modeler*. Furthermore, a 3D graphic line is defined (code line 4–11). The line is black, with a width of 0.002 units and the line is not dashed. The *from_object* of the line (code line 12–14) is defined as a very tiny white cube that is not visible. The *to_object* of the line (code line 15–17) is a plane with a width and height of 0.1 units that takes the image from the “map” variable as texture. Figure 7.9 visualized a 3D instance of an *ARWFMM* “triggers” *relationclass_instance* according to the *VizRep* definition in *VizRep Example 2*.

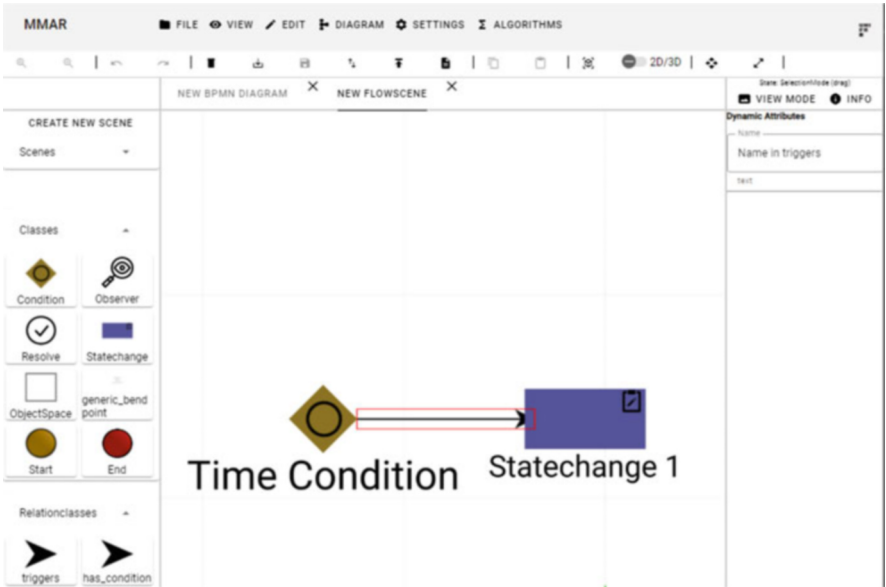


Fig. 7.9 Visualized 3D instance of an *ARWFMM* triggers relationclass_instance according to the *VizRep* definition in *VizRep Example 2*. The relationclass_instance connects two class_instances

VizRep Example 2: Example of the VizRep definition of a “triggers” relationclass

```

1  async function vizRep(gc) {
2      let map = 'data:image/png;...5CYII=';
3      let icon = map;
4      await gc.rel_graphic_line(
5          'black',
6          0.002,
7          false,
8          0,
9          0,
10         0
11     );
12     await gc.rel_from_object(
13         await gc.graphic_cube(0.006, 0.006, 0.006, 'white')
14     );
15     await gc.rel_to_object(
16         await gc.graphic_plane(0.1, 0.1, 'grey', map)
17     );
18 }

```

7.5.5 3D Interaction and Animation Loop

The *M2AR Modeler*’s modeling canvas differs from that of traditional modeling clients. The rendering of the entire 3D environment and interactions within the canvas are more challenging because the modeling canvas is an entire 3D environment. Typically, a 3D scene is rendered at 60 frames per second. However, the *M2AR Modeler* does not require such a high frame rate. It is more important that modeling activities run smoothly on all kinds of devices, such as on 2D desktop and mobile devices, as well as on HMDs. To achieve this, the standard animation loop of the *THREE.js* environment has been modified to render the scene only when there are changes in the scene or the view. This modification enables the client to run on low-end devices that would otherwise be unable to handle a 3D environment.

Interaction in 3D environments differs from traditional 2D desktop interaction. As shown in Fig. 4.22 in Sect. 4.6, mouse-based interaction in 3D environments is based on raycasting. Therefore, the *M2AR Modeler* includes a module for 2D screen interaction with the 3D modeling canvas. This implementation translates 2D screen coordinates to intersection points in the 3D environment. Interaction with the model instance becomes possible, e.g. selecting, moving, rotating, or scaling objects.

7.5.6 Hybrid ARWFMM Implementation

The *M2AR Modeler* is implemented in a generic way to handle as many meta-modeling concepts as possible. However, certain aspects that are specific to certain modeling languages cannot be implemented generically. Thus, the *M2AR Modeler* implements some “hybrid algorithms” that rely still on the concepts of a metamodel but are hard coded in the client code base. Since the introduced ARWFMM is not suitable for 2D modeling environments, we translated the ADOxx implementation into a metamodel based on our new M2AR meta²-model.

The *M2AR Modeler*’s meta²-model implementation currently lacks some of the concepts required for the new modeling language. For example, it does not provide a way to replace standard *VizRep* visualizations, which are general for a class, with a specific 3D representation defined in an *attribute_instance*. This is specifically needed when modeling *Detectables* and *Augmentations* in an *ObjectSpace* model, or when creating *References* to *Detectables* and *Augmentations* in a *Statechange* model. In these cases, we want to model with the specific 3D representation defined in the *attribute_instances* and not with the standard *VizRep* representation defined in the metamodel. Thus, the *M2AR Modeler* implements hybrid algorithms which run only if a user is modeling instances of the *ObjectSpace* or *Statechange* metamodels. The algorithms check then if an alternative 3D visualization is defined in the attached *attribute_instances* and replace the standard *VizRep* visualization with the specific 3D object. Figure 7.10 shows an example of these hybrid algorithms. On the left side, an *ObjectSpace* model is visible that contains two *class_instances*. A *Detectable* and an *Augmentation*. Both instances are visualized with their standard *VizRep* presentation. On the right side, the same *scene_instance* is visible, but the two instances have attached specific *attribute_instance* values, once in the form of a marker image and once in the form of a GLTF file that contains a 3D object of a bed. Since the hybrid algorithms of the *M2AR Modeler* are checking for exactly such *attribute_instances*, the standard *VizRep* visualization is replaced in the 3D modeling environment by the alternative 3D objects.

7.5.7 ARWFMM AR Engine

As elaborated in Chap. 5, there is a need for an AR Engine that is capable of executing models of the ARWFMM modeling method as AR application. Chapter 6 proposed a direct integration of such an engine into the *M2AR Modeler*.

However, in the initial prototypical implementation, the AR Engine was developed independently of the *M2AR Modeler*. In the following, the first implementation of the AR Engine on ADOxx, as well as a second implementation based on M2AR are introduced.

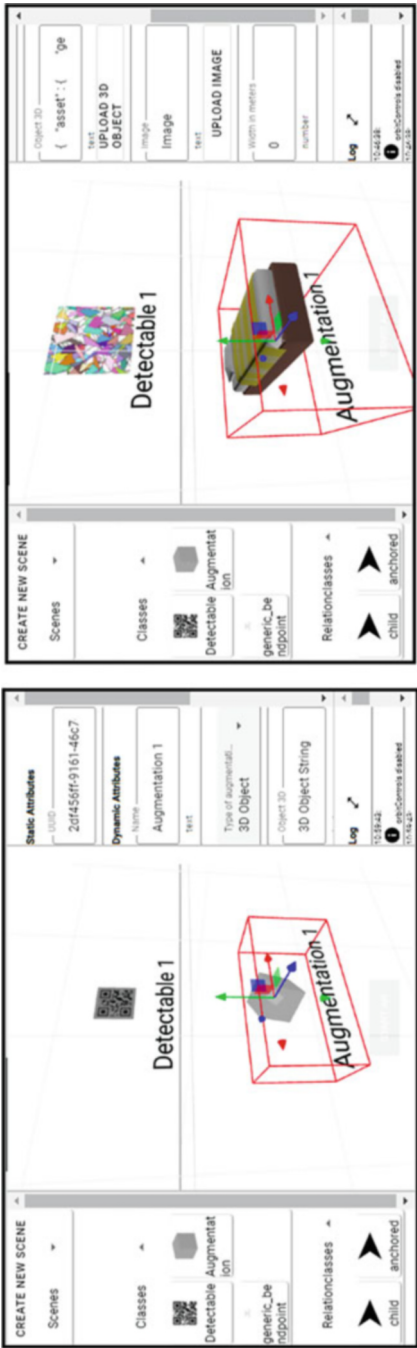


Fig. 7.10 Example of hybrid algorithms replacing standard VizRep visualizations at run-time

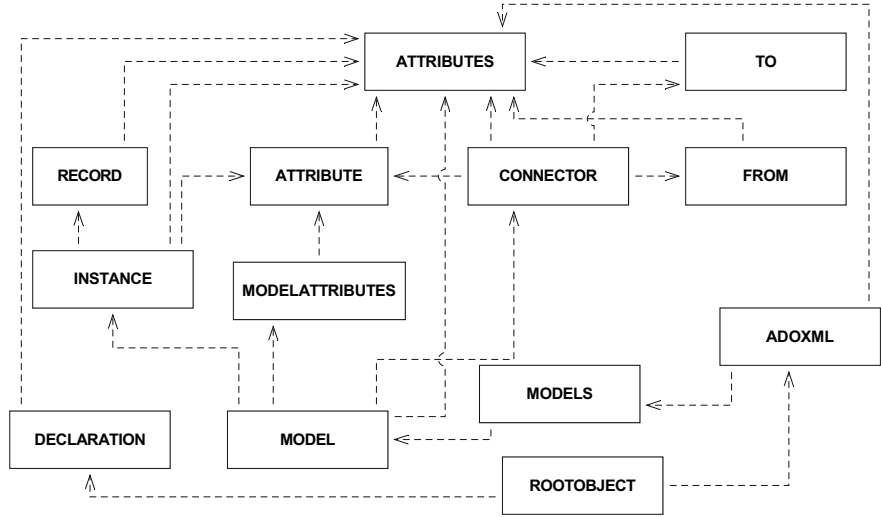


Fig. 7.11 Simplified class diagram representing the ADOxx model structure translated into TypeScript classes

7.5.7.1 AR Engine Implementation Based on ADOxx

The purpose of the first implementation of the *AR Engine* is to transform *ADOxx* models created with the implementation introduced in Chap. 5 into *AR* applications. The technology used for this implementation is exactly the same as that used in the *M2AR Modeler*. This section discusses the first prototype of the *AR Engine* that converts *ARWFMM* models into *AR* applications.

Since the *AR Engine* uses TypeScript as programming language, the *ADOxx* XML model instance structure was translated into a TypeScript class structure. Thus, each *ADOxx* model instance can be parsed into a structure of 13 classes. This facilitates further processing of model instances in TypeScript. Figure 7.11 shows the class diagram that represents the *ADOxx* model structure translated into TypeScript classes. The *ADOxx* XML format does not differentiate between all meta²-concepts, such as classes or relationclasses. It only defines an *INSTANCE* concept, and the parent meta-concept must be derived from the referenced name of the instance.

At the initialization of the *AR Engine*, an *ADOxx* XML file with the model instances representing the *ARWFMM* model must be imported. Then, this file is translated into the TypeScript class structure. In a next step, the different instances are filtered into the different *ARWFMM* concepts. The images to detect and the *3D* objects are loaded into the *3D* environment in a background process.

For generating the *AR* workflow defined by the *ARWFMM* model imported at initialization, the entire workflow defined by the model is compiled into a nodelist, similar to a state machine. The node distinguishes the concept it represents, such as

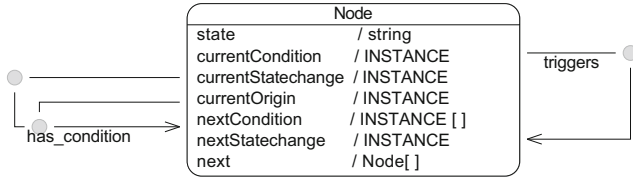


Fig. 7.12 Nodelist structure of the first *AR Engine* for creating *AR* workflows

Origin, *Condition* or *Statechange*, and the next node is referenced according to the *FlowScene* model. Figure 7.12 shows the structure of the nodelist for the *AR Engine*.

Each node in the nodelist has a state attribute that can be set to “active”, “inactive”, or “done”. The type of node and its instance in the *FlowScene* model are distinguished by the attributes “currentOrigin”, “currentCondition” and “currentStatechange”. The property “next” is an array that contains all nodes that should be activated after the current node has been set to “done”. Since a *FlowScene* is not a linear workflow, but can have multiple active conditions or triggered *Statechanges* at the same time, such an approach is necessary. By creating such a nodelist, the entire workflow of *FlowScene* is represented in one single structure that can be traversed easily and fast by the animation loop of the *AR* application.

After the nodelist has been initialized, the first node is set to “active” and the user can start a *WebXR AR* session by clicking a button. The *WebXR* experimental feature “image-tracking” is enabled and all images defined in the *ObjectSpace* model are tracked in every frame. By detecting these images and considering other concepts defined in *Conditions*, e.g., timers, the user is led through the *AR* workflow. Thus, the attributes of activated *Statechanges* are applied to the previously loaded *3D* objects, e.g., visibility, position, and rotation. It should be noted that the *WebXR* “image-tracking” feature is only available on mobile devices and not on *HMDs*. This is due to privacy concerns regarding camera streams.

7.5.7.2 AR Engine Implementation Based on M2AR

As explained in Chap. 5, *ADOxx* is not suitable for modeling *3D* models. Therefore, the *ARWFMM* modeling method has been implemented on the new metamodeling platform. The new *M2AR Modeler* is natively based on a TypeScript class structure, making the translation of XML code into a TypeScript class structure unnecessary. To generate *AR* applications from *ARWFMM* models, some adaptations had to be made to the *AR Engine*. The new model instances are no longer based on the generic structure of *ADOxx*, but on the new *Global Shared Datastructure* introduced in Sect. 7.3. Therefore, all model instances are already typed and do not require filtering during initialization. This simplifies the generation of the nodelist. Furthermore, the new metamodeling platform is based on *UUIDs*, which makes it much easier to identify instances compared to the *ADOxx* implementation, where identifiers are only name strings.

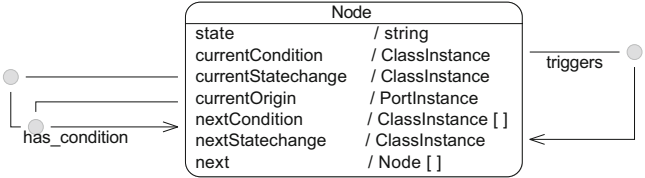


Fig. 7.13 Nodelist structure of the second *AR Engine* for creating *AR* workflows

Figure 7.13 shows the new structure of the nodelist for the second implementation of the *AR Engine*. Thereby, the different properties are not only *INSTANCES*, but are directly typed as provided by the metamodel and the *Global Shared Datastructure*. The general logic behind the second *AR Engine* remains the same, but filtering and traversal is much easier than with the old structure.

The second *AR Engine* was implemented for an evaluation project of *ARWFMM* on the new platform, but has not been directly integrated into the *M2AR* metamodeling platform. It is based on the same technology and structure as the *M2AR Modeler*, so it could be integrated as a module in the *M2AR Modeler*, as proposed in Fig. 7.1. A demonstration of the use of the *AR Engine* will be shown in Chap. 8.

7.6 Metamodeling Client

The *M2AR Metamodeling Client* is currently undergoing active implementation, with a focus on ensuring compatibility by adopting a consistent technology stack. This strategic decision to use a uniform set of technologies is driven by the need to achieve seamless integration and interoperability within the proposed architecture. By aligning the technological basis of the *Metamodeling Client* with that of existing *M2AR* modules, we are able to facilitate a more cohesive and efficient development process, and it may even be possible to combine the different modules into one combined *M2AR* metamodeling web client.

7.7 Collaboration Server

The collaboration server is not yet implemented in the metamodeling platform. There have been test implementations to prove the concept. Thus, the collaboration server will be implemented in a client-server approach using the *WebSocket* protocol⁸ for communication between the central server and the connected *M2AR Modelers*.

⁸ <https://github.com/websockets/ws> last visited on: 01.03.2024.

7.8 Need for a First Evaluation

In this chapter, we introduced the first prototype of the proposed *M2AR* metamodeling platform as a [DSR](#) artifact. As required by the [DSR](#) research methodology, a rigorous evaluation of design science artifacts is essential (Peffers et al. [2012](#)). Thus, in the next chapter, we will conduct a first evaluation of the artifacts presented.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



Chapter 8

Evaluation of the M2AR Platform Prototype



Some parts of this chapter have been published in a similar form as a research paper in: 2024 ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS) with the title: *M2AR: A Web-based Modeling Environment for the Augmented Reality Workflow Modeling Language* (Muff and Fill 2024b) and in: Conceptual Modeling - 43rd International Conference (ER) with the title: *Multi-Faceted Evaluation of Modeling Languages for Augmented Reality Applications - The Case of ARWFML* Muff and Fill (2024c)

This chapter evaluates three different aspects of the *M2AR* platform. First, it includes a comparative evaluation of the global generic- and specific requirements (see Chaps. 3 and 4), against the first implementation of the metamodeling platform *M2AR* (see Chaps. 6 and 7), and against the first implementations of the *ARWFMM* language implemented on *M2AR*. This is followed by a demonstration of *M2AR* and its *ARWFMM* implementation, and third, an empirical evaluation of the comprehensibility of the *ARWFMM* and its language concepts.

8.1 Evaluation Against Requirements

Chapters 3 and 4 introduced the different *global generic requirements* and *global specific requirements*. There are a total of 31 requirements against which to evaluate the introduced *M2AR* meta²-model, the *ARWFMM*, and the prototype of the *M2AR* metamodeling platform. This section briefly discusses each requirement and analyzes whether it has been addressed in the current implementation.

GGR1 states that in addition to hand gestures and voice control, other interaction options adapted for virtual and augmented reality should also be enabled. Since the *THREE.js* and *WebXR* implementation support different interaction possibilities for *VR* and *AR*, this requirement has been partially met. However, the prototypical implementation of *M2AR* does not yet make use of these possibilities.

GGR2 states that it must be possible to attach virtual objects to real objects by means of anchoring, i.e., reference point information must be stored in some way in the model. This requirement is met, since there are specific concepts that consider relative positioning and anchoring in the *M2AR* meta²-model, as well as in the *ARWFMM* and the *M2AR* modeling platform.

GGR3 states that object recognition during run-time must be enabled. This requirement is not met, since, at the time of writing, it is not allowed to access camera feeds of *AR* devices during run-time, due to privacy reasons. Conceptually this is considered in the *ARWFMM*. That is why the requirement is considered to be partially met.

GGR4 states that accurate positioning of both the user and objects in the physical setting must be determined. This requirement is met, as the technical implementation with *THREE.js* and *WebXR* allows the tracking of users in the environment.

GGR5 states that it must be possible to detect real objects to achieve an overlay of real and the virtual objects. Due to the privacy reasons mentioned, real dynamic object detection is not possible at the moment, but it is possible to detect surfaces, which allows for a partial fulfillment of the requirement.

GGR6 states that semantic inferences of the user's context must be possible. This has been addressed in Chap. 4.5 and in Muff and Fill (2022a). This concepts could be directly mapped to the architecture of the *M2AR* platform on a technical level, e.g., by feeding information in the provided import/export formats in JSON notation and/or by using the component for hybrid algorithms. However, the concept has not yet been addressed in the *M2AR* implementation. Thus, the requirement is only partially fulfilled.

GGR7 states that real-time collaboration must be possible. This requirement is considered in the conceptual design of the *M2AR* metamodeling platform. However, it is currently in development and not yet integrated.

GGR8 states that models must support different views of the same model for different situations. At the moment, this is considered by allowing the user to view a model in a *2D* view or a *3D* view in the *M2AR* client. Thus, the requirement is met.

GGR9 states that the connection of related models and the appropriate visualization of these connections must be enabled. At the moment, this is not considered in any implementation. Thus, the requirement is not met. Since we are proposing a metamodeling platform, different languages are supported and it is conceptually possible to interconnect them, e.g., as demonstrated in the *ARWFMM* implementation. Therefore, this topic is subject to future research.

GSR1–GSR5 are requirements that consider coordinate systems, including positioning and rotations. The different coordinate systems, as well as position and rotations, have been considered in the *M2AR* meta²-model, and in the *M2AR* implementation of the *M2AR* metamodeling platform. Furthermore, positions and rotations are also part of the *ARWFMM*. Thus, **GSR1** to **GSR5** are met.

GSR6–GSR10 are requirements considering visualization. With the introduction and implementation of the *VizRep* (see Sects. 6.3.1.1 and 7.5.4), and the consider-

ation of the concept in the *M2AR* meta²-model, including the possibility to use the *GLTF* data format, the requirements **GSR6** to **GSR10** are met.

GSR11 states that a metamodeling environment must enable the real-time tracking of a user's position and orientation relative to the base coordinate system. Since the implementation is based on *THREE.js* and *WebXR*, this requirement is met.

GSR12–GSR15 are requirements considering the detection of *2D* images and *3D* objects. Both of these requirements are met by considering the concepts in the *ARWFMM*. Due to the mentioned privacy issues for the *AR* devices, the dynamic detection of *3D* objects is not possible in the current implementation with *THREE.js* and *WebXR*. For the detection of *2D* images the *WebXR API* provides an experimental feature that has been used for the implementation of the *AR Engine* (see Sect. 7.5.7). However, this feature is currently only available for smartphones and tablets, but not for *HMDs*. Thus, **GSR12** and **GSR13** are fully met, and **GSR14** and **GSR15** are partially met.

GSR16 states that a metamodeling environment should allow the tracking of human bodies and their specific parts. Since different body tracking features are directly integrated into the *WebXR API* in the implementation of the *M2AR* metamodeling platform, this requirement is met. However, there was not yet a specific focus on this topic.

GSR17 states that a metamodeling environment must allow the use of raycasting concepts. Since the implementation is based on *THREE.js* and *WebXR*, this requirement is met.

GSR18–GSR19 ask for mapping real-world objects without knowing the exact visual representation and the inference of context information. This has not been addressed in the implementation of the *M2AR* prototype. Thus, **GSR18** and **GSR19** are not met.

GSR20 states that a metamodeling environment must enable adapted interaction approaches to define metamodels. At the moment, metamodels are defined in JSON format. The *Metamodeling Client* is in development, but not yet integrated into the platform. Therefore, **GSR20** is not met.

GSR21 states that a metamodeling environment must enable adapted interaction approaches for interaction with models or model environments. As described in **GGR1**, the *THREE.js* and *WebXR* implementation supports different interaction possibilities for *VR* and *AR*. However, the prototypical *M2AR* implementation currently implements only mouse-based interaction. Thus, **GSR21** is only partially met.

GSR22 states that a metamodeling environment must enable real-time synchronization of virtual information embedded in virtual and augmented reality environments. This corresponds to **GGR7**. Thus, **GSR22** is not met.

Table 8.1 summarizes all global and specific requirements with the description of the requirement and if the requirement is covered.

Table 8.1 List of generic and specific requirements for joining metamodeling with extended reality, summarizing if the requirement is covered in the current implementation. ✓ means that the requirements is covered, (✓) means partially covered, and ✗ means not covered

Short	Requirement	Covered?
GGR1	In addition to hand gestures and voice control, other interaction options adapted for virtual and augmented reality should also be enabled	(✓)
GGR2	It must be possible to attach virtual objects to real objects by means of anchoring, i.e., reference point information must be stored somehow in the model	✓
GGR3	Object recognition during run-time must be enabled, e.g., by using machine learning algorithms for object recognition, as well as semantic reasoning	(✓)
GGR4	The accurate positioning of both the user and objects in the physical setting must be determined, including indoor and outdoor environments	✓
GGR5	To achieve an overlay of the real and virtual objects and to annotate them with additional information, the detection of real objects must be possible	(✓)
GGR6	Based on the real environment, semantic inferences, i.e., states and consequences about the user's context must be possible. Possibilities to enable this are, e.g., ontological or rule-based reasoning	(✓)
GGR7	Real-time collaboration should be supported, whether modeling in a multi-user environment in the same location or over a distance	✗
GGR8	Models must support different views of the same model for different situations	✓
GGR9	The connection of related models and the appropriate visualization of these connections must be enabled	✗
GSR1	A metamodeling environment must support 3D coordinates for the base coordinate system	✓
GSR2	A metamodeling environment must support 3D relative coordinates	✓
GSR3	A metamodeling environment must support absolute coordinates	✓
GSR4	A metamodeling environment must support 3D coordinates for positioning	✓
GSR5	A metamodeling environment must support 3D rotations	✓
GSR6	A metamodeling environment must allow for the visualization of 3D objects	✓
GSR7	A metamodeling environment must allow the use of well-known 3D data formats	✓
GSR8	A metamodeling environment must allow for the definition of 3D visualization on the level of metamodels	✓
GSR9	A metamodeling environment must allow dynamic changes of 3D visualizations	✓
GSR10	A metamodeling environment must support concepts to define the dynamic behavior of model components on the level of metamodels	✓

(continued)

Table 8.1 (continued)

Short	Requirement	Covered?
GSR11	A metamodeling environment must enable the real-time tracking of a user’s position and orientation relative to the base coordinate system	✓
GSR12	A metamodeling environment must support concepts for the detection of 2D images	✓
GSR13	A metamodeling environment must support concepts for the tracking of 2D images	✓
GSR14	A metamodeling environment must support concepts for the detection of 3D objects	(✓)
GSR15	A metamodeling environment must support concepts for the tracking of 3D objects	(✓)
GSR16	A metamodeling environment should allow for the tracking of human bodies and their specific parts	✓
GSR17	A metamodeling environment must allow the use of raycasting concepts	✓
GSR18	A metamodeling environment must consider concepts for the mapping to real-world objects without knowing the exact visual representation	✗
GSR19	A metamodeling environment must allow for the inference of context information	✗
GSR20	A metamodeling environment must enable adapted interaction approaches for defining metamodels	✗
GSR21	A metamodeling environment must enable adapted interaction approaches for interaction with models or model environments	(✓)
GSR22	A metamodeling environment must enable real-time synchronization of virtual information embedded into virtual or augmented reality environments	✗

8.2 Demonstration of the M2AR Implementation

This section demonstrates the ARWFMM implementation on M2AR, as well as use case examples on the M2AR Modeler and the AR Engine.

8.2.1 ARWFMM Implementation on the M2AR Metamodeling Platform

For a first evaluation of the prototypical implementation of the M2AR metamodeling platform, the ARWFMM has been implemented on M2AR. In this section, we will demonstrate this ARWFMM implementation, since it uses most of the features we considered for the implementation of the M2AR metamodeling platform. In the following, we will discuss the implemented metamodels, as well as the adapted notation of the ARWFMM on M2AR and a language overview.

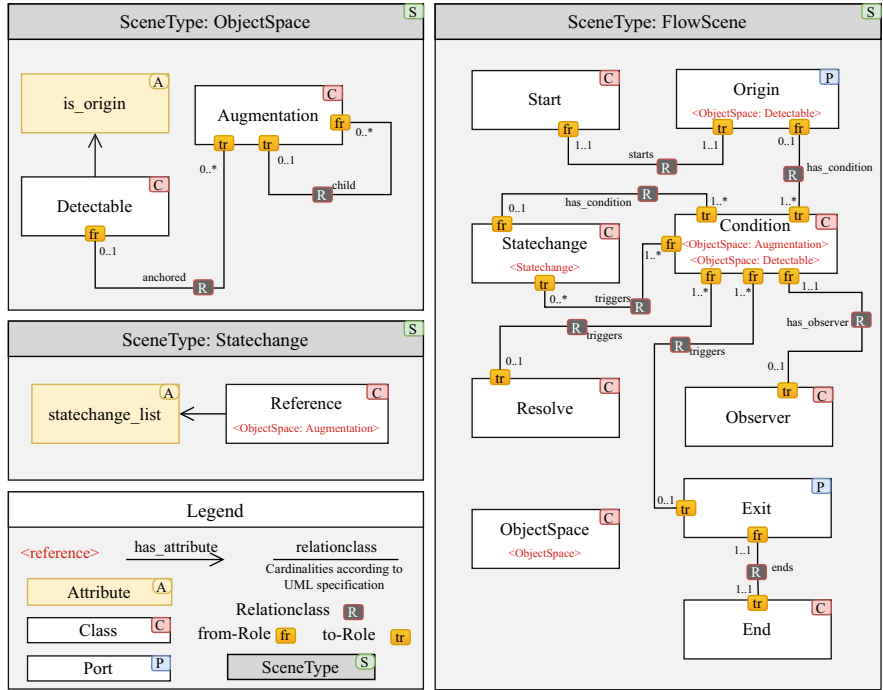


Fig. 8.1 Reduced ARWFMM metamodel for *ObjectSpace*, *Statechange*, and *FlowScene* on the M2AR metamodeling platform

8.2.1.1 ARWFMM Metamodels on M2AR

Figure 8.1 shows the three metamodels created as *SceneType* on the M2AR metamodeling platform. The three metamodels *ObjectSpace*, *Statechange*, and *FlowScene* are depicted in a reduced form. Not all attributes are shown, since this would make the metamodels very hard to read. In the following, we briefly go through the new metamodel and highlight differences to the metamodel introduced in Chap. 5. An *ObjectSpace* consists of two main classes: *Augmentation* and *Detectable*. An *Augmentation* is virtual data that can be used in AR, e.g., a 3D object or a text label. *Augmentations* can be linked to other *Augmentations* via the “child” relationclass.

As introduced in Sect. 6.1.1, *relationclasses* are not connected directly to *classes*, but to a *from_role* and a *to_role*. These *roles* are then connected to other concepts, i.e., *classes*. Thus, all the *relationclasses* in Fig. 8.1 connect to two *roles* denoted as *fr* and *tr*.

In addition to the “child” *relationclass*, *Augmentations* can be linked to *Detectables* through the “anchored” *relationclass*. *Detectables* have, among others, an attribute “is_origin” to indicate whether they are a surrogate for the AR application’s world-origin. A *Statechange* model describes changes in the appearance of

Augmentations defined in the *ObjectSpace* model, such as visibility, position, or rotation. The *Statechange* model includes only one *class*, *Reference*, which lists the changes (*statechange_list*) that occur to a referenced *Augmentation*. This model is essential for representing dynamic changes in the AR environment. Note, that the concept *<reference>* on *classes* and *ports* in the metamodels is corresponding to the meta-layer concept for *attribute_types* referencing a *role*. Thus, if there is a *<reference>* on a *class* or *port*, this means that the *class* or *port* has an *attribute* with an *attribute_type* referencing a *role* for connecting other concepts—see Sect. 6.1.1.6.

The *FlowScene* metamodel includes a *Start* and an *End* *class*, and contains a reference to an instance of the *ObjectSpace* model (*ObjectSpace class*). In contrast to the first implementation on *ADOxx*, the *ObjectSpace class* has an *Origin port* that refers to a *Detectable* in the *ObjectSpace* model. The *FlowScene* metamodel also includes *Statechanges* and *Resolves*, which are triggered by *Conditions*. Furthermore, *Conditions* can trigger upon observing *Augmentations* or *Detectables*, or via the connected *Observer* concept. *Statechanges*, *Conditions*, and *Resolves* are interconnected and determine how the resulting application should respond to specific conditions in the AR environment, guiding its workflow and interactions.

The *ObjectSpace*, *Statechange*, and *FlowScene* metamodel files corresponding to the M2AR meta²-model contain 738, 583, and 1441 lines of code, respectively. These files are quite large. However, with the introduction of the *Metamodeling Client* that is currently in development, the specification of metamodels will become much easier and users will not have to deal with JSON files.























8.2.1.2 ARWFMM 3D Notation on M2AR

In addition to the changes in the ARWFMM metamodels, the M2AR metamodeling platform also requires the change of the ARWFMM language notation in comparison to the *ADOxx* implementation introduced in Chap. 5. In contrast to *ADOxx*, the M2AR metamodeling platform is 3D-based. Furthermore, many of the visualizations created with the *VizRep* (see Sect. 7.5.4) are dynamically changing during run-time. Thus, the notation of the ARWFMM has been adapted to meet the new requirements for 3D-based modeling. Table 8.2 shows the 3D notation of the ARWFMM modeling language concepts. For each *SceneType*, the visual notation is shown in the 3D Notation column. If there is a dynamic visualization depending on *attribute_instance* values, an example visualization is shown in the *Dynamic Example* column. For a semantic description of the different concepts, refer to Table 5.2.

8.2.1.3 ARWFMM Language Overview on M2AR

Figure 8.2 provides a language overview by displaying a subset of the different models needed for modeling with ARWFMM. The *ObjectSpace* model (top left) defines

Table 8.2 3D notation of the ARWFMM modeling language. For each *SceneType*, the visual notation is shown in the *3D Notation* column. If there is a dynamic visualization depending on *attribute_instance* values, an example visualization is shown in the *Dynamic Example* column

	Concept	3D Notation	Dynamic Example		Concept	3D Notation	Dynamic Example
ObjectSpace	Detectable			FlowScene	Origin		
	Augmentation				Start		
	Anchored				End		
	Child				Statechange		
	Reference				Starts		
FlowScene	Condition			FlowScene	Has Condition		
	Resolve				Triggers		
	Observer				Has Observer		
	Exit				Ends		
	ObjectSpace						

nine *Detectable* instances as image marker representations and six *Augmentation* instances as 3D objects. The *Statechange* area (top right) displays two examples of *Statechange* model instances that define the appearance of the referenced *Augmentations* when triggered. The *FlowScene* (bottom) defines the actual path of the AR workflow. It references a *Detectable* as the world-origin of the AR application, as well as different *Conditions* for detecting markers 2–9 (*Detectables*). Furthermore, it defines *Statechange* instances, referencing *Statechange* models to be triggered.

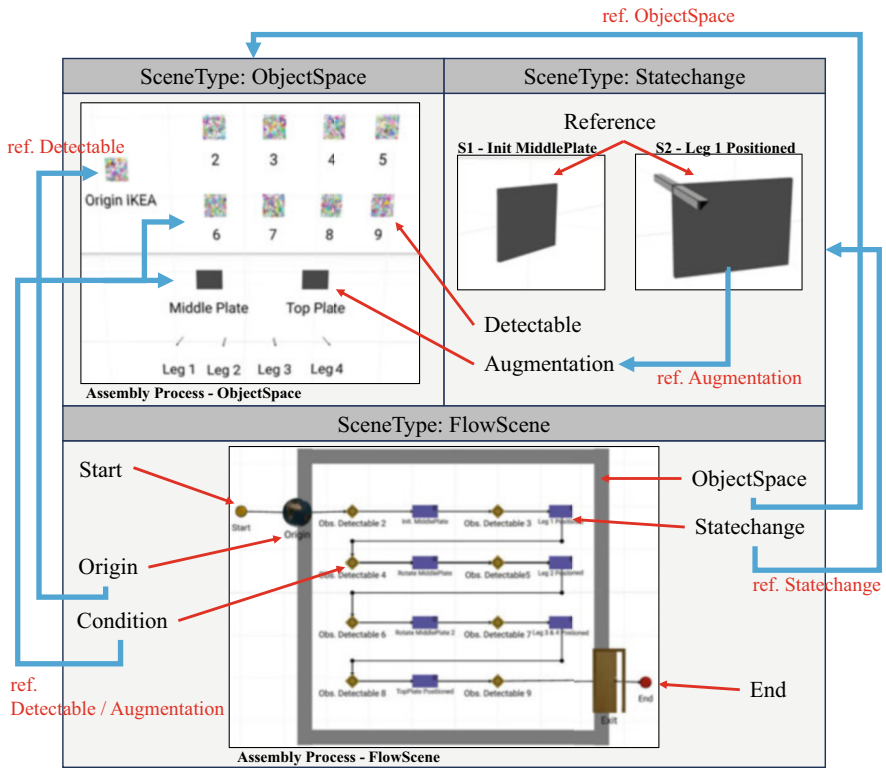


Fig. 8.2 ARWFMM language overview, demonstrated in an example of a furniture assembly process on the new *M2AR* metamodeling platform

The following sections provide use case examples for modeling with the *M2AR Modeler* and executing the modeled *AR* applications in the *AR Engine* to demonstrate the functionality of the *ARWFMM* implementation on *M2AR*.

8.2.2 Furniture Assembly Use Case

The purpose of the first use case is to guide the user through the assembly process of a bedside table in *AR*, cf. Chap. 5. As described in the modeling procedure of *ARWFMM* in Sect. 5.3.4, it is proposed to start by creating an empty *ObjectSpace*. This is followed by the creation of the *FlowScene* and the placeholders for the application workflow. Figure 8.3 shows the *FlowScene* model of the use case. The *FlowScene* contains an *ObjectSpace* referencing the already created empty *ObjectSpace* model. Furthermore, it contains an *Origin Port* that will later reference a *Detectable* from the *ObjectSpace* model. The model then defines the workflow

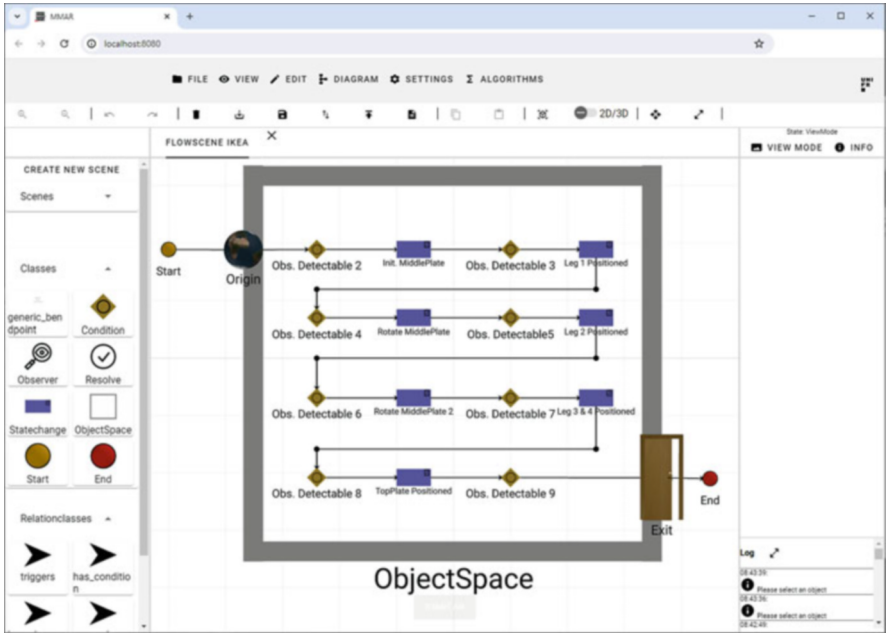


Fig. 8.3 Screenshot of the *FlowScene* of the IKEA use case in the *M2AR Modeler*

of the [AR](#) application. In this case, it is a linear workflow with *Conditions* always triggering a *Statechange*. All of the defined *Conditions* in the use case are referencing a *Detectable* in the *ObjectSpace* model. However, these *Detectables* must first be defined.

Figure [8.4](#) shows a screenshot of the *M2AR Modeler* while modeling the *ObjectSpace* of the use case. The *ObjectSpace* model contains ten *Detectables*. Every *Detectable* has a unique image as visual representation, coming from the *Image attribute_instance*. One of the *Detectables*, the “Origin IKEA” is marked as “origin” and is referenced by the *Origin Port* of the *FlowScene* model. The other *Detectables* are referenced by the different *Conditions* in the *FlowScene* model. Furthermore, the *ObjectSpace* model defines six different *Augmentations*, one for each part of the furniture piece of the real world. These *Augmentations* are visualized according to the *3D* visualization uploaded as *GLTF* file to the *Object 3D attribute_instance* of the *Augmentations*. Lastly, for each *Statechange* in the *FlowScene* model, a *Statechange* model is created in the *M2AR Modeler*. Figure [8.5](#) shows two screenshots of the *M2AR Modeler* while modeling *Statechange* models for the first *Statechange* “Init MiddlePlate” (left) and the second *Statechange* “Leg 1 Positioned” (right). In both models the *3D* view is used. The *Statechange* models contain *Reference class_instances* that reference to *Augmentations* of the *ObjectSpace* model. In the left *Statechange* only one *Reference* is used that references the “Middle Plate” *Augmentation* of the *ObjectSpace* model. The right

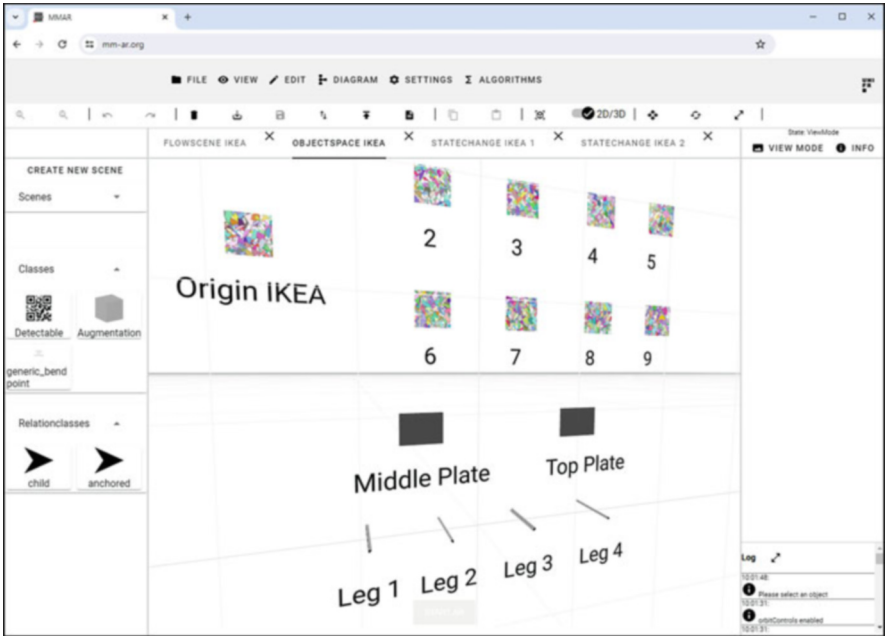


Fig. 8.4 Screenshot of the *ObjectSpace* of the IKEA use case in the *M2AR Modeler* in the 3D view

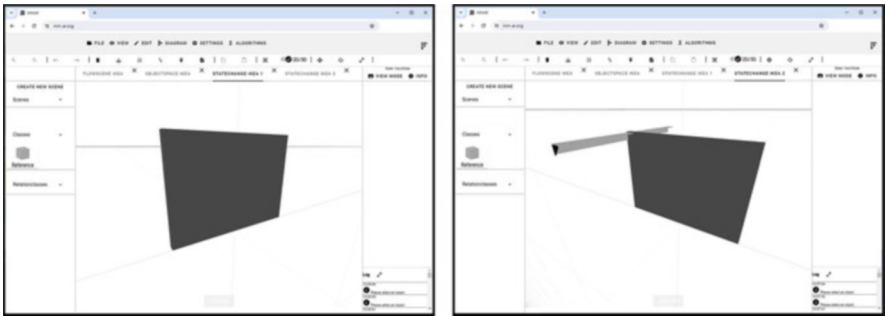


Fig. 8.5 Screenshots of two *Statechanges* of the IKEA use case in the *M2AR Modeler* in the 3D view

Statechange references also the “Middle Plate” *Augmentation*, and additionally the “Leg 1” *Augmentation*.

By setting the *attribute_instances* of the different *References*, as well as by positioning, scaling, and rotating the different *References* that are visualized according to the referenced *Augmentations*, one can define how the AR application shall visualize the different *Augmentations* when a *Statechange* is triggered. After initially modeling the different models, in this use case an *ObjectSpace*, a *FlowScene*, and

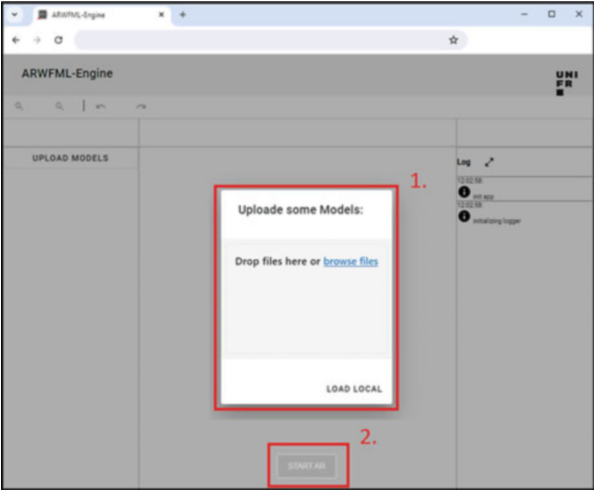


Fig. 8.6 Screenshot of the AR Engine web client

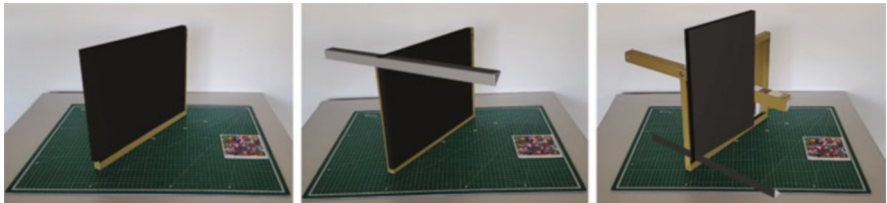


Fig. 8.7 Screenshots of the IKEA use case in the AR Engine

seven *Statechanges*, the nine models can be exported as one JSON file through a button in the *M2AR Modeler*. Figure 8.6 shows a screenshot of the *AR Engine* web client in the browser. The *AR Engine* is accessible on any WebGL-enabled browser device supporting the WebXR device API, e.g., smartphones, tablets, or AR HMDs. In the *AR Engine* one can upload the previously exported JSON file with an upload form (1) and the AR application can then be started with a click on the “START AR” button (2)—see Fig. 8.6.

As soon as the AR session is loaded, the *AR Engine* checks for the detection of the origin. After the origin detection, the AR workflow starts according to the defined *FlowScene*. Figure 8.7 shows three screenshots of the running *AR Engine*, taken on a *Samsung Galaxy Tab S7* tablet, executing the IKEA use case defined above. The left image shows the state of the AR application after the *Statechange* “Init MiddlePlate” has been executed. Furthermore, the middle image shows the application state after the execution of *Statechange* “Leg 1 Positioned” and the right image after the execution of *Statechange* “Rotate MiddlePlate”.

8.2.3 Machine Process Use Case

The second use case involves a work process that is enhanced with AR. For example, imagine a scenario where a user needs to cut five equal metal pieces using a special cutting machine. The process is simple: Start the machine, insert the raw metal piece, and close the cover. The machine will automatically reopen once the piece is finished, allowing the user to repeat the process as many times as needed. Finally, the machine must be stopped. To instruct novice users of the machine with AR, a very simple workflow has been modeled with the ARWFMM. The goal is to start the machine, cut five pieces, and stop the machine. The machine has a small display on the front, on which different states can be visualized. In this case, an image of an origin marker is visualized. The objective of the AR application is to guide the user to start and end the machine by indicating the correct button to push at the right moment in the process. Initially, an arrow is displayed, followed by the text label “Push here!” after three seconds. When starting, the machine continuously sends the machine state and the number of pieces cut to an API that is accessible through the company network. As soon as five pieces have been cut, the AR application visualizes the arrow, and after three seconds the text label is displayed at the position of the button to stop the machine.

As introduced in the modeling procedure, first an empty *ObjectSpace* model is created, and the raw workflow of the AR application is modeled in a *FlowScene* model. Figure 8.8 shows a screenshot of the *FlowScene* model in the M2AR

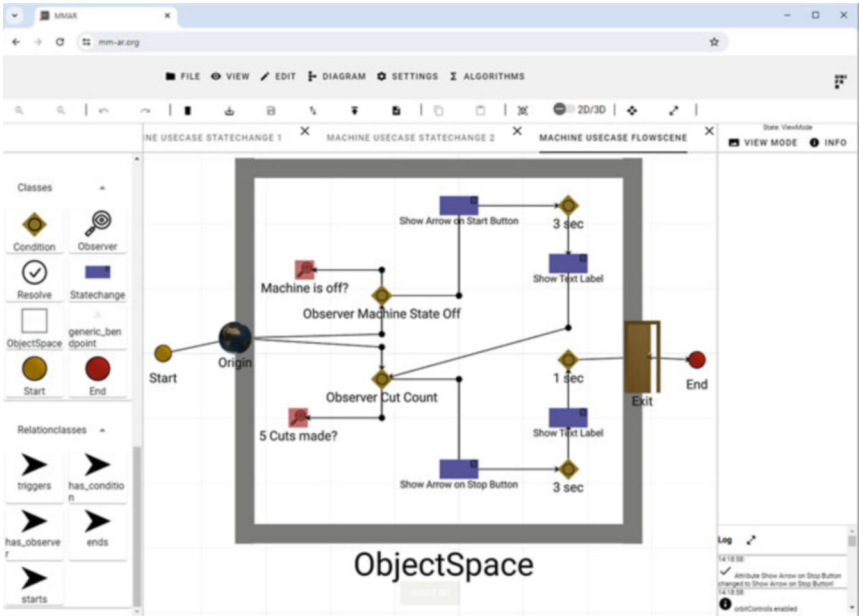


Fig. 8.8 Screenshot of the *FlowScene* of the machine process use case in the M2AR Modeler

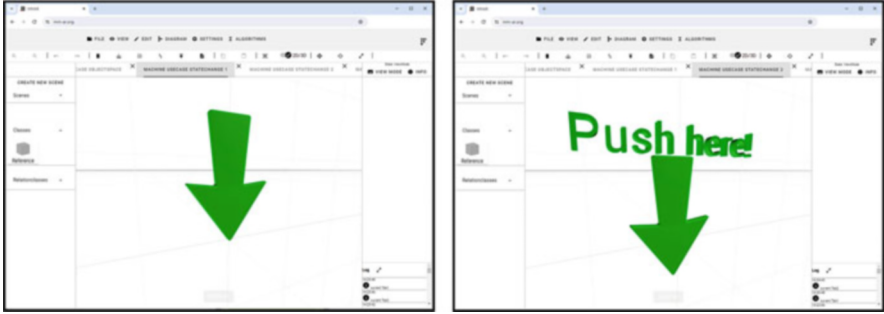


Fig. 8.9 Screenshots of two *Statechanges* of the machine process use case in the *M2AR Modeler* in the *3D* view

Modeler. The workflow is slightly more complex than the workflow shown in Sect. 8.2.2. In this use case, the *Origin* port has two *Conditions* attached. Both *Conditions* have an *Observer* attached to them. The first *Observer* is checking if the machine is off via the *API* accessible over the network. If the machine is off, the condition is triggered and the first two *Statechanges* “Show Arrow on Start Button” and “Show Text Label” are activated—see *Statechange* models in Fig. 8.9. The left image shows the first *Statechange* and the right image shows the second *Statechange*. After this second *Statechange*, the second *Condition* “Observer Cut Count” is activated if this is not already the case. This second *Condition* also has an *Observer* attached which checks for the number of cut pieces over the *API*. As soon as five pieces are cut, the *Condition* triggers the *Statechanges* “Show Arrow on Stop Button” and after three seconds the “Show Text Label” which visualize the arrow and the text label at the position of the button to stop the machine.

To allow for these visualizations, the initially created *ObjectSpace* must be extended with a *Detectable* for the origin marker and the two *Augmentations* for the arrow and the text label visualizations. Figure 8.10 shows a screenshot of this *ObjectSpace* model in the *M2AR Modeler*.

Again, the final model can be exported as JSON file to run on an *AR* device with the *AR Engine*. The *AR Engine* does not support the execution of *AR* applications depending on *Observer* calls. Thus, Fig. 8.11 only shows an example of how the generated *AR* application could look.

8.2.4 Office Tour Use Case

The third use case involves an office tour of a research center that houses different research groups. The purpose of the *AR* application is to allow the user to randomly explore the research center and display visual information about the different research groups and their locations in the *AR* application. This is the first use case

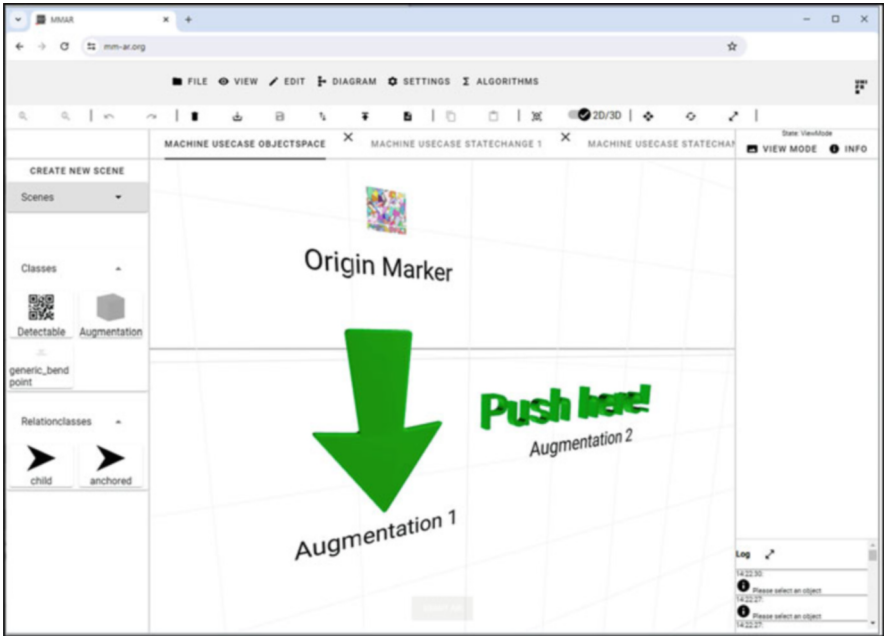


Fig. 8.10 Screenshot of the *ObjectSpace* of the machine process use case in the *M2AR Modeler* in the *3D* view



Fig. 8.11 Example of the machine process use case in the *AR Engine*

that does not specify a predefined process, but rather many different possible next steps. The use case involves 12 research groups in total.

As shown in Fig. 8.12, the *ObjectSpace* includes one origin *Detectable* and one *Detectable* for each research group. Each *Detectable* has attached (*anchored*) an *Augmentation*, which is visualized as a *3D* cube with a texture map describing the research group’s activities as text.

In addition, the *FlowScene* for the office tour use case defines 12 *Conditions* that are all directly connected to the *Origin* port—see Fig. 8.13. Each *Condition* references a *Detectable* marker from the *ObjectSpace* for triggering the next *Statechange*. All *Conditions* are activated from the beginning. Each *Condition* is followed by a separate *Statechange* that refers to a *Statechange* model, e.g. as shown

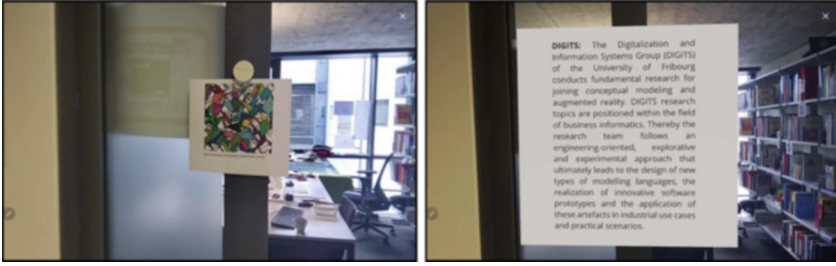


Fig. 8.15 Example of the office tour use case in the *AR Engine*

serve as examples for both the *Business Process Perspective* (refer to Sect. 3.1.4) and the *IT Perspective* (refer to Sect. 3.1.5). For instance, the use cases presented to address server issues or the process support use case (see Chap. 3) can be modeled similarly to the use cases demonstrated in this section. However, the *ARWFMM* currently lacks the ability to reference other modeling languages and utilize knowledge from referenced models directly in the *AR* application, e.g., for reasoning purposes. Additionally, *VR/AR*-assisted modeling, as introduced in the use case for the *Strategic Perspective* (refer to Sect. 3.1.3), is currently not feasible.

The next section provides a more empirical evaluation of the *ARWFMM* by assessing its comprehensibility.

8.3 Empirical Evaluation of *ARWFMM* Concepts on *M2AR*

As demonstrated in Sect. 8.2, the *ARWFMM* modeling method has been implemented on the new *M2AR* metamodeling platform. This section is empirically evaluating the comprehensibility of the *ARWFMM* and its language concepts on *M2AR*. Since *ARWFMM* is a visual modeling language, the concepts and its understanding have been evaluated as they are visually represented in the *M2AR* implementation.

This evaluation has been done as a separate DSR project and defined its own specific research questions. Thus, the following research question, denoted as *Evaluation Research Questions* (ERQs) was investigated:

- ERQ1: To what extent are the newly introduced *Augmented Reality Workflow Modeling Method* (*ARWFMM*) and its associated language concepts comprehensible to users with varying levels of expertise in conceptual modeling and augmented reality?

For this evaluation, the empirical evaluation of the *AR Engine* and the *M2AR* metamodeling platform have been left out, since we first wanted to evaluate the use of the modeling language. This empirical evaluation would follow in the next *DSR* iteration.

8.3.1 Study Design

To address the *evaluation research questions*, our study employs a between-subject design (Charness et al. 2012), in which two user groups were independently tested. We selected this design because we wanted to identify differences between user groups with different backgrounds. The main threat to the validity for this set up are discussed in Sect. 8.3.5. The experiments took place in November and December 2023.

Each study comprised two parts: (1) An introduction to the ARWFMM, covering the basic concepts of AR, the specific ARWFMM language concepts, and an introduction to the new M2AR Modeler. The introduction was held in a 45 minute interactive presentation with intermediate examples on the different language concepts on the new M2AR Modeler. (2) The participants were asked to complete a questionnaire regarding their demographics, understanding of ARWFMM models, and general understanding of ARWFMM language concepts. Model understanding was evaluated by showing the participants five sections of screenshots of the M2AR Modeler with three different possibilities for each resulting AR workflow in the AR Engine. The participants had to choose for each question the correct result. Figure 8.16 shows an example of a question about model understanding. The goal of these first five questions (Q1–Q5) was to assess the understanding of the visual models and to determine if the result of the provided models is predictable for the participants. In addition, participants were presented with statements regarding the ARWFMM modeling method and asked to identify all correct statements in five separate questions—see general questions (GQ1–GQ5) in Table 8.3. The purpose of the general questions was to gather information about specific concepts of the modeling method. GQ1 aimed to determine if participants could recall the different ARWFMM SceneTypes that were presented in a selection of other SceneTypes that were not related to ARWFMM. GQ2 aimed to determine if participants could recall the purpose of the three different SceneTypes by selecting the true statements about each one. Questions GQ3 to GQ5 were separate for each SceneType of ARWFMM and presented participants with various choices of different concepts. Participants were required to select only the concepts that are part of the SceneType to which the question refers. The purpose of these questions is to assess participants' ability to recall the different concepts from the various SceneTypes. The main goal of this first survey was to test the understanding of the modeling method.

In the following sections, we discuss the experimental subjects, evaluation metrics, and quantitative results of the questionnaires.

8.3.2 Experimental Subjects

To recruit subjects with at least minimal modeling knowledge, we used university courses in “Introduction to Business Informatics” and “Databases” at the University

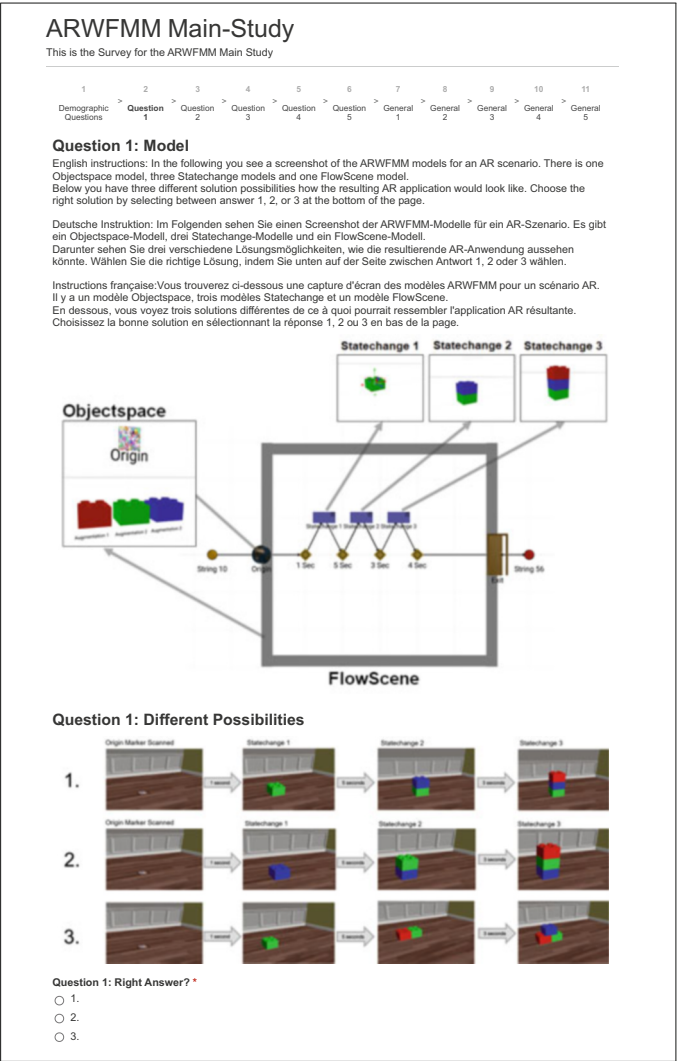


Fig. 8.16 Example of a survey question in the model understanding part of the study

of Fribourg. Our overall sample consisted of 35 students, all of them in an age range between 18 and 44, with most participants below 24 (see Table 8.4). The study included high school graduates (25), individuals with Bachelor’s degrees (7), Master’s degrees (2), as well as Doctoral degrees (1)—see Table 8.5.

To survey the participants familiarity with *conceptual modeling* and *augmented reality*, we collected data based on five-point *Likert* scales (Likert 1932). Table 8.6 shows the participants’ self-assessed familiarity with the fields of conceptual modeling and augmented reality. Responses are categorized as “very familiar” “familiar,”

Table 8.3 General questions (GQ1–GQ5) for evaluating ARWFMM method understanding

GQ	Question	Purpose
GQ1	Which are the 3 different types of models that can be created with ARWFMM?	SceneTypes
GQ2	ARWFMM defines the 3 types ObjectSpace, Statechange, and FlowScene. Choose the 3 right statements	General statements
GQ3	Which concepts are available in a ObjectSpace model? Choose the 2 available modeling concepts of an ObjectSpace model	ObjectSpace understanding
GQ4	Which concepts are available in a Statechange model? Choose the only available modeling concepts of a Statechange model	Statechange understanding
GQ5	Which concepts are available in a FlowScene model? Choose the 7 introduced modeling concepts of a FlowScene model	FlowScene understanding

Table 8.4 Age distribution of participants

Age category	# Participants
18–24	24
25–34	10
35–44	1
Σ	35

Table 8.5 Educational attainment of participants in the study

Level of education?	# Participants
High school graduate	25
Bachelor’s degree	7
Master’s degree	2
Doctoral degree	1
Σ	35

Table 8.6 Self-reported familiarity with conceptual modeling and augmented reality

Familiarity with:	Conceptual modeling?	AR?
Very familiar	0	0
Familiar	4	3
Somewhat familiar	14	7
Somewhat unfamiliar	9	8
Very unfamiliar	8	17
Σ	35	35

“somewhat familiar,” “somewhat unfamiliar,” and “very unfamiliar”, with corresponding participant counts for each category. The data indicate that there is a range of exposure and understanding among the participants in these two domains. Out of the 35 respondents, 18 rated their familiarity with conceptual modeling as “somewhat familiar” or higher, with four considering themselves “familiar”. For augmented reality, 25 participants rated their familiarity as “somewhat unfamiliar” or “very unfamiliar”.

8.3.3 Evaluation Metrics

Measuring the comprehensibility of a modeling method is more challenging. This is because modeling is typically performed using a modeling tool, making it difficult to separate the method from the tool for evaluation. The *Methods Evaluation Model* by Moody (2003) defines four measures that need to be considered when evaluating an *Information Systems* design method. *Performance*, *Perceptions*, *Intentions*, and *Behavior*. Since we wanted to objectively evaluate only the modeling method and the method is not yet used in practice, we only evaluated *Performance*, which is composed by the measures *Actual Efficiency* and *Actual Effectiveness* (Kekes 1994). The efficiency of a method is determined by the amount of effort required to apply it. The effectiveness of a method is determined by how well it achieves its objectives. To assess the performance of an unproductive method, we have reduced the method to assessing whether participants can distinguish between different model scenarios and whether they understand the different language concepts, by showing them different models for validation (see Fig. 8.16), and by asking questions about the methodology in general.

8.3.4 Results

This section presents the results of the user study.

ERQ1: *Comprehensible of the ARWFMM Modeling Method:* Table 8.7 quantifies the percentage of correct answers for each of the five questions on model understanding (Q1 to Q5) in the two studies with a total of 35 participants. The data is presented to illustrate the percentage of participants choosing the correct answer out of three given propositions, e.g., as visible in Fig. 8.16. A “Total” row aggregates the performance across both studies, providing an overall success rate for each question.

The results indicate high total percentage of correct responses for Q1 (0.94), Q4 (0.94), and Q5 (0.97) on average, suggesting a robust model understanding. Q2 had moderate accuracy (0.91) with a notable increase in the second study group. Q3 had the lowest average accuracy (0.83), which is still moderate. These findings suggest that participants had a good understanding of the models shown and the desired AR application scenario resulting from the models.

Table 8.8 shows the average accuracy of responses across the two studies for the five general questions on ARWFMM method understanding (GQ1–GQ5)—see questions in Table 8.3. The questions were in a multiple-choice format with varying

Table 8.7 Accuracy of responses across the two studies for the five questions on model understanding (Q1 to Q5)

	Q1	Q2	Q3	Q4	Q5
Study 1	0.94	0.88	0.81	0.94	1.00
Study 2	0.95	0.95	0.84	0.95	0.95
Total	0.94	0.91	0.83	0.94	0.97

Table 8.8 Average correct answer rate across the two studies for the five questions on ARWFMM method understanding (GQ1 to GQ5)

	GQ1	GQ2	GQ3	GQ4	GQ5
Study 1	0.91	0.78	0.90	0.92	0.84
Study 2	0.88	0.87	0.92	0.86	0.79
Total	0.89	0.83	0.91	0.89	0.81

Table 8.9 Correlation coefficients between correct answer rates for the different ARWFMM questions of the study and familiarity measures for conceptual modeling or augmented reality

	Q1–Q5	GQ1	GQ2	GQ3	GQ4	GQ5
Conceptual modeling	0.02	0.39	0.32	−0.12	0.07	0.33
Augmented reality	−0.11	0.31	0.08	0.11	0.22	0.19

numbers of choices and correct options. The bottom row displays the total average value for correct answers over all participants. The five questions provide insights into the relative difficulty or clarity of each concept of the modeling language.

Both studies showed that question GQ3 (concepts of *ObjectSpace*) had the highest and second-highest average for correct answers, with scores of 0.90 in the first study and 0.92 in the second study, indicating that it was likely the easiest or most clearly understood language concept. The lowest average correct answer rate in the first study was for GQ2 (general statements on the three *SceneTypes*) at 0.78, while in study two, it was for GQ5 (*FlowScene* concepts) at 0.79, suggesting that these concepts were the most difficult or least clear. In study two, the mean correct answer rates for questions GQ2 and GQ3 were generally higher compared to study one, but lower for questions GQ1, GQ4, and GQ5. The total average correct answer rate across both studies for each question shows that overall, GQ3 had the highest average correct answer rate (0.91), while GQ2 had the lowest (0.83), reinforcing the notion that GQ3 was the best understood or easiest language concept, and GQ2 was one of the most challenging or least clear.

To address performance differences with respect to the different levels of familiarity in conceptual modeling or AR, we have calculated the correlation coefficients between the familiarity measures and the correct answer rates for the different ARWFMM questions. The correlation for age and the level of education was not considered, since most of the participants were in the same categories. None of the correlation coefficients for familiarity with conceptual modeling or augmented reality, based on the five-point Likert scale, showed a clear correlation. As visible in Table 8.9, eight out of 12 values showed a correlation coefficients between −0.3 or 0.3, which, according to Hinkle et al. (2003), is considered as *negligible correlation*. Only four values were slightly out of this range, which is still considered as low correlation. It must be noted that the sample size of the study was only 35, which could affect the robustness and generalizability of the correlation analysis findings.

Overall, these results provide insights into which aspects of the ARWFMM may require clearer explanations or better instructional materials. The relative difficulty or ease perceived by the participants in answering the questions indicates

areas that need improvement. In summary, the language concepts and models created with **ARWFMM** are highly comprehensible. All questions regarding model understanding were answered correctly in over 80% of cases. Furthermore, an average correct answer rate of above 0.8 in eight out of ten areas, and over 0.9 in four areas for general language understanding was achieved. There was no discernible correlation between different levels of familiarity with augmented reality or conceptual modeling and the correct answers given in the questionnaire. This suggests that deep knowledge in these areas is not necessary for understanding the **ARWFMM** and its models. Thus, **ERQ1** can be answered as follows:

The newly introduced **ARWFMM** and its associated language concepts are in general highly comprehensible for users with various levels of knowledge in augmented reality and conceptual modeling.

8.3.5 *Threats to Validity*

When evaluating the proposed methodology and its implementation, we conducted a controlled experiment to ensure internal validity, while also taking into account external validity. It is essential to recognize the limitations of our study design. Further experimentation and validation are necessary before considering the application of our methodology in real-world scenarios.

In addressing external validity, we recruited a substantial number of participants with at least a basic understanding of modeling. Due to the experiment taking place within the confines of a university course, it is possible that the results may not fully reflect the complexities and variables of real-world applications. Furthermore, the study's use cases were predetermined, restricting participants' ability to create their own models. This limitation may impact the applicability of our findings to a wider range of diverse and unplanned scenarios.

Additionally, evaluating the methodology through screenshots rather than direct interaction within the tool is also a limitation, as it may not fully capture the depth of understanding that could be achieved through interactive use. Moreover, the study's focus on relatively simple and small-scale scenarios raises concerns about the generalizability of the findings to more complex and varied real-world applications.

Regarding internal validity, the use of screenshots to test model understanding could be seen as a deviation from measuring the true performance of the modeling method, especially since it is integrated with a **3D** modeling tool. The decision was a trade-off between a very controlled setup for precise measurement and a more open setup that could have introduced additional variables. Direct interaction with models within the tool in the study could have provided a more authentic experience for participants, but this could have introduced biases in measuring their understanding.

Finally, participant selection in our studies may introduce bias due to varying levels of technical proficiency and familiarity with modeling methods and augmented reality technologies. Additionally, the sequence of introducing the prototype and modeling method before administering the questionnaire could have influenced participants' perceptions and understanding. Although the potential learning effect is an important factor to consider, the interdependence between the modeling language and the web application may help to reduce its influence.

8.4 Summary of the M2AR Platform Prototype Evaluation

In this section, we evaluated parts of the research artifacts related to this book. Initially, we evaluated the artifacts against global requirements. Out of the 31 generic or specific requirements, our implementation meets 18 (58.1%), partially meets 7 (22.5%), and does not meet 6 (19.4%). Therefore, more than 80% of all requirements are at least partially met.

In a second step, we demonstrated the introduced metamodeling platform on three use cases. This showed that the prototype is functional and can be used for modeling in 2D and in 3D. Furthermore, the use cases showed that it is possible to model use cases as derived in Chap. 3 with the prototypical *M2AR* metamodeling platform and the *ARWFMM* implementation. What is missing so far is the capability to model in VR or AR, which excludes use cases such as the use case for AR BMC modeling as introduced in Sect. 3.1.3 and VR/AR-assisted modeling in general.

Finally, we evaluated the general *ARWFMM* language and model understanding in two user studies. The results indicate that the *ARWFMM* and its associated language concepts are highly comprehensible (ERQ1). The evaluation of the usability of the *ARWFMM* and the *M2AR Modeler* was not considered in this study and will be the subject of future research.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



Chapter 9

Summary and Outlook



The last chapter of this book summarizes the completed work and provides concluding remarks. Section 9.1 aligns the project results with the research questions. The limitations of the research, as well as future work are discussed in Sect. 9.2. Finally, Sect. 9.3 closes this work with a summary.

9.1 Alignment with the Research Questions

This section is devoted to addressing the initially defined research questions:

RQ1: “What are the necessary components and concepts in virtual and augmented reality in general and in the context of metamodeling?”: With the introduction to extended reality in Sect. 2.2, the main concepts for virtual and augmented reality in general have been introduced. Furthermore, by deriving 31 generic or specific requirements for joining metamodeling with extended reality in the area of three-dimensional coordinate mappings, visualization of 3D model components, detection and tracking, context information, or interaction, the second part of the research question has been answered.

RQ2: “What components of a meta²-model must be considered to allow the integration of metamodeling for 3D environments in extended reality?”: By discussing the implications of these requirements in the different areas of metamodeling in Sect. 4.7, the second research question has been answered.

RQ3: “How can an existing meta²-model be adapted or extended to incorporate 3D and XR features to meet emerging requirements, or what characteristics would define a newly developed meta²-model enriched with 3D and XR capabilities?”: By discussing existing metamodeling platforms in Sect. 5.2.2 and by proposing the new *M2AR* meta²-model in Sect. 6.1.1, the third research question has been answered. Since 3D features are required on a very high level, i.e., on the

meta²-level, a new meta²-model was introduced instead of extending an existing one.

RQ4: “What are the architectural components of a 3D enhanced metamodeling platform that considers extended reality?”: By conceptually proposing a new 3D enhanced metamodeling platform that considers extended reality in Chap. 6, the fourth research question has been answered.

RQ5: “How can a 3D enhanced metamodeling platform considering extended reality be technically realized and evaluated?”: The final research question has been answered by discussing the prototypical implementation of the new *M2AR* metamodeling platform (Chap. 7), demonstrating it on use cases (Sect. 8.2), and evaluating the comprehensibility of its Augmented Reality Workflow Modeling Method (*ARWFMM*) implementation in an empirical user study (Sect. 8.3).

9.2 Limitations and Future Research

This work comes with some limitations. First, this research is considered as ground research with the goal of discovering a very large research area. This comes with the drawback of being potentially less specific. Although we tried to cover the whole topic as well as possible, there are areas where we only scratched the surface in this book. For example, in the second half of the book we focused more on the area of *knowledge-based VR/AR*, including design-time and run-time aspects. We did not discuss in detail the area of *VR/AR-assisted modeling*, including specific interaction possibilities in *XR*, as well as the automated elicitation of conceptual models within *VR* or *AR*. This is also a significant research area that we plan to explore in the future. Furthermore, we did not discuss certain areas in detail, such as considerations for outdoor *XR* applications, open 3D standards, or privacy and security aspects of extended reality applications. All of these are important and complex topics that require further investigation. Furthermore, it is important to note that the literature analysis covers only the period from 2000 to the first half of 2022. However, we regularly monitor new developments in the field and are confident that the findings of this analysis remain valid. Finally, the empirical evaluation conducted in this work only considers the comprehensibility of the *ARWFMM* on the prototypical implementation of *M2AR*. It does not empirically evaluate the usability of the software artifact, as the *M2AR* implementation is only a first prototype.

In future research, our aim is to further investigate the area. This includes improving the *M2AR* metamodeling platform and the *ARWFMM*. It would be interesting to explore approaches for 3D enhanced modeling in *VR* or *AR*, or to incorporate real-world references such as 3D environment scans, or 360° videos or images into the modeling environment. Furthermore, as discussed in Chap. 7, the *M2AR Metamodeling Client* and the *Collaboration Server* are currently in development and will further improve the *M2AR* metamodeling platform.

9.3 Summary

In this work, it has been investigated how to combine extended reality with conceptual modeling, and particularly metamodeling. To gain an initial understanding of the field, the two main areas, i.e., modeling and extended reality have been introduced, and an extensive literature review on related work in 3D, VR, and AR, in combination with conceptual modeling and metamodeling has been conducted. Additionally, a morphological analysis has been used to derive use cases and generic requirements for extended reality applications in combination with metamodeling. Thereby, the areas of *VR/AR-assisted modeling* and *knowledge-based VR/AR* have been discovered, distinguishing between *design-time* and *run-time* scenarios. Based on the use cases found, and the generic requirements derived, specific requirements have been derived for joining metamodeling with extended reality. This includes, for example, requirements for three-dimensional coordinate mappings, visualization of 3D model components, detection and tracking, context information, or interaction. Based on the derived use cases and the resulting requirements for combining XR with metamodeling on a general level, the book delved deeper into the area of *knowledge-based VR/AR*, i.e., the specific area of generating augmented reality applications with the help of metamodeling by proposing ARWFMM, a domain-specific visual modeling method for creating augmented reality applications. This, in addition to the already discovered requirements, exposed the need for a 3D enhanced metamodeling platform considering extended reality. Such a platform, the M2AR metamodeling platform, has been conceptualized and prototypically implemented with state-of-the-art 3D web technology. Furthermore, the platform was evaluated by comparing it with the initial requirements derived, demonstrating its use cases, and conducting an empirical user study to assess the considerable comprehensibility of the ARWFMM.

Although a comprehensive view of the area was attempted, some questions remain unanswered. This includes in particular the further research concerning not met requirements like the semantic inference of context information, 3D object recognition, real-time collaboration, multi-view approaches, or model interconnection. Furthermore, it appears that knowledge-based VR/AR is an area worth closer investigation.

In conclusion, this book demonstrates that traditional approaches for modeling are insufficient when bringing the new paradigm of extended reality to the well-established area of metamodeling, as existing metamodeling platforms only consider two dimensions. Therefore, the introduction of XR to metamodeling requires a new 3D-enhanced metamodeling environment, such as the proposed M2AR metamodeling platform.

Considering again Fig. 1.1 in Chap. 1, we can draw conclusion on how the combination of metamodeling with extended reality, as part of design-oriented business informatics, can bridge the gap between the economic objectives of business administration and the technical possibilities of computer science, thereby

supporting the realization of economic goals, such as increasing market share and profit, or advancing sustainability, efficiency, and quality.

The application of methodologies such as **ARWFMM** can facilitate the development of extended reality applications, thereby reducing costs and increasing efficiency. For instance, the implementation of large-scale training applications in **XR** could enhance sustainability, efficiency, and consequently, profitability. This is because individuals can learn how to perform their duties in a flexible manner, regardless of their location, rather than having to travel to a specific place to be instructed by a professional instructor. Another illustrative example of significant interest is the use of **XR** applications to elicit enterprise models. Such applications have the potential to improve the efficiency of model creation and documentation in various ways, thus reducing costs. For example, they can be used to automatically elicit enterprise models via sensor data, or they can be used to create immersive real-world models. Furthermore, the facilitated development of such applications will result in an increased production of virtual content, which in turn will enhance the technology of **XR** in general. This can represent a self-reinforcing cycle that supports digitalization in general.

In the author's opinion, extended reality has the potential to advance conceptual modeling and metamodeling, bringing us closer to the vision of integrating conceptual modeling into daily work and life—see Sect. 1.1. Therefore, research in the area of **XR** conceptual modeling should continue in the future. This includes research in the areas of knowledge-based **VR/VR**, and **VR/VR**-assisted modeling, as well as in the area of **3D** enhanced metamodeling environments.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



Appendix A

ARWFMM Use Case Example with FDMM

The full list of the FDMM formalism of the use case example can be found in Muff ([2024](#)).

Bibliography

- Ahmadyan, Adel, Liangkai Zhang, Artsiom Ablavatski, Jianing Wei, and Matthias Grundmann. 2021. Objectron: a large scale dataset of object-centric videos in the wild with pose annotations. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7818–7827. <https://doi.org/10.1109/CVPR46437.2021.00773>.
- AREA. 2022. AREA AR use case descriptions. Technical report. Augmented Reality for Enterprises Alliance (AREA). Accessed November 07, 2023. <https://thearea.org/wp-content/uploads/2022/05/AR-Use-Case-Descriptions-and-Examples.pdf>.
- Arfken, George B. 1985. *Mathematical Methods for Physicists*. 3rd ed. Academic Press. ISBN 978-0-12-059820-5. <https://doi.org/10.1016/C2013-0-10310-8>.
- Azuma, Ronald T. 1993. Tracking requirements for augmented reality. *Communications of the ACM* 36 (7): 50–51. ISSN 0001-0782. <https://doi.org/10.1145/159544.159581>.
- Azuma, Ronald T. 1997. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments* 6 (4): 355–385. <https://doi.org/10.1162/pres.1997.6.4.355>.
- Betz, Stefanie, Daniel Eichhorn, Susan Hickl, Stefan Klink, Agnes Koschmider, Yu Li, Andreas Oberweis, and Ralf Trunko. 2008. 3D representation of business process models. In *Modellierung betrieblicher Informationssysteme - Modellierung zwischen SOA und Compliance Management - 27-28. November 2008 Saarbrücken*, ed. Peter Loos, Markus Nüttgens, Klaus Turowski, and Dirk Werth, LNI, vol. P-141, 73–87. GI. <https://dl.gi.de/handle/20.500.12116/23619>.
- Blei, David M. 2012. Probabilistic topic models. *Communications of the ACM* 55 (4): 77–84. <https://doi.org/10.1145/2133806.2133826>.
- Blei, David M., and John D. Lafferty. 2005. Correlated topic models. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5–8, 2005, Vancouver]*, 147–154. <https://proceedings.neurips.cc/paper/2005/hash/9e82757e9a1c12cb710ad680db11f6f1-Abstract.html>.
- Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3: 993–1022. ISSN 1532-4435. <https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf>.
- Blinn, James F., and Martin E. Newell. 1976. Texture and reflection in computer generated images. *Communications of the ACM* 19 (10): 542–547. ISSN 0001-0782. <https://doi.org/10.1145/360349.360353>.
- Bock, Alexander C., and Ulrich Frank. 2021. Low-code platform. *Business & Information Systems Engineering* 63 (6): 733–740. <https://doi.org/10.1007/s12599-021-00726-8>.
- Booth, Andrew, Anthea Sutton, and Diana Papaioannou. 2016. *Systematic Approaches to a Successful Literature Review*. 2nd ed. Sage. ISBN 978-1-4739-1245-8.

- Borcard, Daniel. 2022. A Comparison of Rules Engines for Constraint Checking in a Node.js-based Metamodeling Platform. Bachelor Thesis. <https://doi.org/10.5281/zenodo.10830792>.
- Borgida, Alexander. 1990. Knowledge representation, semantic data modelling: What's the difference? In *Proceedings of the 9th International Conference on Entity-Relationship Approach (ER'90)*, 8–10 October, 1990, Lausanne, ed. Hannu Kangassalo, 1. ER Institute. <https://www.sigmod.org/publications/dblp/db/conf/er/Borgida90.html>.
- Bork, Dominik, and Hans-Georg Fill. 2014. Formal aspects of enterprise modeling methods: a comparison framework. In *47th Hawaii International Conference on System Sciences, HICSS 2014, Waikoloa, HI, January 6–9, 2014*, 3400–3409. IEEE Computer Society. <https://doi.org/10.1109/HICSS.2014.422>.
- Bork, Dominik, and Ben Roelens. 2021. A technique for evaluating and improving the semantic transparency of modeling language notations. *Software and Systems Modeling* 20 (4): 939–963. <https://doi.org/10.1007/s10270-021-00895-w>.
- Boyd-Graber, Jordan, David Mimno, and David Newman. 2014. Care and feeding of topic models: problems, diagnostics, and improvements. In *CRC Handbooks of Modern Statistical Methods*. CRC Press. ISBN 9780429098826. https://home.cs.colorado.edu/~jbg/docs/2014_book_chapter_care_and_feeding.pdf.
- Brambilla, Marco, Jordi Cabot, and Manuel Wimmer. 2017. *Model-Driven Software Engineering in Practice*. 2nd ed. Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers. <https://doi.org/10.2200/S00751ED2V01Y201701SWE004>.
- Brunschwig, Lea, Ruben Campos-Lopez, Esther Guerra, and Juan de Lara. 2021. Towards domain-specific modelling environments based on augmented reality. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, Madrid, ES, May 2021, 56–60. IEEE. ISBN 978-1-66540-140-1. <https://doi.org/10.1109/ICSE-NIER52604.2021.00020>.
- Buchmann, Robert Andrei, and Dimitris Karagiannis. 2015. Agile modelling method engineering: lessons learned in the comvantage research project. In *The Practice of Enterprise Modeling - 8th IFIP WG 8.1. Working Conference, PoEM 2015, Valencia, November 10–12, 2015, Proceedings*, ed. Jolita Ralyté, Sergio España, and Oscar Pastor, vol. 235. Lecture Notes in Business Information Processing, 356–373. Springer. https://doi.org/10.1007/978-3-319-25897-3_23.
- Buchmann, Robert Andrei, Dimitris Karagiannis, and Niksa Visic. 2013. Requirements definition for domain-specific modelling languages: the comvantage case. In *Perspectives in Business Informatics Research - 12th International Conference, BIR 2013, Warsaw, September 23–25, 2013. Proceedings*, ed. Andrzej Kobylinski and Andrzej Sobczak, vol. 158. Lecture Notes in Business Information Processing, 19–33. Springer. https://doi.org/10.1007/978-3-642-40823-6_3.
- Bühlmann, Marcel. 2023. Realizing Domain-Specific Mechanisms and Algorithms on the MM-AR Metamodeling Platform. Master's thesis, University of Fribourg. <https://doi.org/10.5281/zenodo.10839131>.
- Cabot, Jordi, and Antonio Vallecillo. 2022. Modeling should be an independent scientific discipline. *Software and Systems Modeling* 21 (6): 2101–2107. <https://doi.org/10.1007/s10270-022-01035-8>.
- Campos-López, Rubén, Esther Guerra, and Juan de Lara. 2021. Towards automating the construction of augmented reality interfaces for information systems. In *Information Systems Development: Crossing Boundaries between Development and Operations (DevOps) in Information Systems (ISD2021 Proceedings)*, Valencia, September 8–10, 2021, ed. Emilio Insfrán, Silvia Abrahão, Marta Fernández, Chris Barry, Michael Lang, Henry Linger, and Christoph Schneider. Universitat Politècnica de València. <https://aisel.aisnet.org/isd2014/proceedings2021/hci/6>.
- Chang, Jonathan, Jordan Boyd-Graber, Sean Gerrish, Chong Wang, and David M. Blei. 2009. Reading tea leaves: how humans interpret topic models. In *NIPS'09 Conference*, 288–296. ISBN 9781615679119. <https://proceedings.neurips.cc/paper/2009/file/f92586a25bb3145facd64ab20fd554ff-Paper.pdf>.

- Charness, Gary, Uri Gneezy, and Michael A. Kuhn. 2012. Experimental methods: between-subject and within-subject design. *Journal of Economic Behavior & Organization* 81 (1): 1–8. ISSN 0167-2681. <https://doi.org/10.1016/j.jebo.2011.08.009>.
- Chen, Peter P. 1976. The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems* 1 (1): 9–36. <https://doi.org/10.1145/320434.320440>.
- Chen, Ding-Yun, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. 2003. On visual similarity based 3D model retrieval. *Computer Graphics Forum* 22 (3): 223–232. <https://doi.org/10.1111/1467-8659.00669>.
- Chen, Peng, Xiaolin Liu, Wei Cheng, and Ronghuai Huang. 2017. A review of using Augmented Reality in Education from 2011 to 2016. In *Innovations in Smart Learning*, 13–18. https://doi.org/10.1007/978-981-10-2419-1_2.
- Churchill, Rob, and Lisa Singh. 2022. The evolution of topic modeling. *ACM Computing Surveys* 54 (10s): 215:1–215:35. <https://doi.org/10.1145/3507900>.
- Cipresso, Pietro, Irene Alice Chicchi Giglioli, Mariano Alcañiz Raya, and Giuseppe Riva. 2018. The past, present, and future of virtual and augmented reality research: a network and cluster analysis of the literature. *Frontiers in Psychology* 9: 2086. ISSN 1664-1078. <https://doi.org/10.3389/fpsyg.2018.02086>.
- Cohen, Jacob. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20 (1): 37–46. <https://doi.org/10.1177/001316446002000104>.
- Corley, Jonathan, and Eugene Syriani. 2014. A cloud architecture for an extensible multi-paradigm modeling environment. In *Joint Proceedings of MODELS 2014 Poster Session and the ACM Student Research Competition (SRC) Co-located with the 17th International Conference on Model Driven Engineering Languages and Systems (MODELS 2014)*, Valencia, September 28–October 3, 2014, ed. Stefan Sauer, Manuel Wimmer, Marcela Genero, and Shaz Qadeer, vol. 1258. CEUR Workshop Proceedings, 6–10. CEUR-WS.org. <https://ceur-ws.org/Vol-1258/poster2.pdf>.
- Crevoiserat, Sophie, Fabian Muff, and Hans-Georg Fill. 2023. Towards augmented reality applications for it maintenance tasks based on ArchiMate models (short paper). In *Companion Proceedings of the 42nd International Conference on Conceptual Modeling: ER Forum, 7th SCME, Project Exhibitions, Posters and Demos, and Doctoral Consortium co-located with ER 2023, Lisbon, November 06–09, 2023*, ed. Claudenir M. Fonseca, José Borbinha, Giancarlo Guizzardi, David Aveiro, Sotirios Liaskos, C. Maria Keet, Estefanía Serral, Fernanda Baião, João Araújo, Tiago Prince Sales, Miguel Mira da Silva, Sergio de Cesare, H. Sofia Pinto, Ladjel Bellatreche, and Simon Hacks, vol. 3618. CEUR Workshop Proceedings. CEUR-WS.org. <https://ceur-ws.org/Vol-3618/pd%5Fpaper%5F5.pdf>.
- Dalton, Jeremy, and Jonathan Gillham. 2019. Seeing is believing. Accessed March 18, 2024. <https://www.pwc.com/gx/en/industries/technology/publications/economic-impact-of-vr-ar.html>.
- de Souza Cardoso, Luís Fernando, Flávia Cristina Martins Queiroz Mariano, and Ezequiel Roberto Zorzal. 2020. A survey of industrial augmented reality. *Computers & Industrial Engineering* 139. <https://doi.org/10.1016/j.cie.2019.106159>.
- Di Ruscio, Davide, Dimitrios S. Kolovos, Juan de Lara, Alfonso Pierantonio, Massimo Tisi, and Manuel Wimmer. 2022. Low-code development and model-driven engineering: two sides of the same coin? *Software and Systems Modeling* 21 (2): 437–446. <https://doi.org/10.1007/s10270-021-00970-2>.
- Dix, Alan, Janet E. Finlay, Gregory D. Abowd, and Russell Beale. 2003. *Human-Computer Interaction*. 3rd ed. Prentice-Hall, Inc. ISBN 0130461091.
- Doerner, Ralf, Wolfgang Broll, Paul Grimm, and Bernhard Jung, eds. 2022. *Virtual and Augmented Reality (VR/AR): Foundations and Methods of Extended Realities (XR)*. Springer International Publishing. ISBN 978-3-030-79061-5. <https://doi.org/10.1007/978-3-030-79062-2>.
- Fettke, Peter. 2006. State-of-the-Art des State-of-the-Art: Eine Untersuchung der Forschungsmethode „Review“ innerhalb der Wirtschaftsinformatik. *WIRTSCHAFTSINFORMATIK* 48 (4): 257. ISSN 0937-6429, 1861-8936. <https://doi.org/10.1007/s11576-006-0057-3>.

- Fill, Hans-Georg. 2009. *Visualisation for Semantic Information Systems*. Gabler. <https://doi.org/10.1007/978-3-8349-9514-8>.
- Fill, Hans-Georg. 2011. Using semantically annotated models for supporting business process benchmarking. In *Perspectives in Business Informatics Research - 10th International Conference, BIR 2011, Riga, October 6–8, 2011. Proceedings*, ed. Janis Grabis and Marite Kirikova, vol. 90. Lecture Notes in Business Information Processing, pp 29–43. Springer. https://doi.org/10.1007/978-3-642-24511-4_3.
- Fill, Hans-Georg, and Dimitris Karagiannis. 2013. On the conceptualisation of modelling methods using the ADOxx meta modelling platform. *Enterprise Modelling and Information Systems Architectures—International Journal of Conceptual Modeling* 8 (1): 4–25. <https://doi.org/10.18417/emisa.8.1.1>.
- Fill, Hans-Georg, and Fabian Muff. 2023. Visualization in the era of artificial intelligence: experiments for creating structural visualizations by prompting large language models. *CoRR*. abs/2305.03380. <https://doi.org/10.48550/ARXIV.2305.03380>.
- Fill, Hans-Georg, and Fabian Muff. 2024. Bridging the mental and the physical world: conceptual modeling and augmented reality. In *Informing Possible Future Worlds. Essays in Honour of Ulrich Frank*, ed. Stefan Strecker and Jürgen Jung. Berlin: Logos Verlag. ISBN 9783832557683. <https://doi.org/10.30819/5768>.
- Fill, Hans-Georg, Timothy Redmond, and Dimitris Karagiannis. 2012a. FDMM: a formalism for describing ADOxx meta models and models. In *ICEIS 2012 - Proceedings of the 14th International Conference on Enterprise Information Systems, Volume 3, Wroclaw, 28 June–1 July, 2012*, ed. Leszek A. Maciaszek, Alfredo Cuzzocrea, and José Cordeiro, 133–144. SciTePress. <https://doi.org/10.5220/0003971201330144>.
- Fill, Hans-Georg, Timothy Redmond, and Dimitris Karagiannis. 2012b. Formalizing meta models with FDMM: the ADOxx case. In *Enterprise Information Systems - 14th International Conference, ICEIS 2012, Wroclaw, June 28–July 1, 2012, Revised Selected Papers*, ed. José Cordeiro, Leszek A. Maciaszek, and Joaquim Filipe, vol. 141. Lecture Notes in Business Information Processing, 429–451. Springer. https://doi.org/10.1007/978-3-642-40654-6_26.
- Fill, Hans-Georg, Susan Hickl, Dimitris Karagiannis, Andreas Oberweis, and Andreas Schoknecht. 2013. A formal specification of the Horus modeling language using FDMM. 11. Internationale Tagung Wirtschaftsinformatik, Leipzig, February 27–March 1, 2013, 73.
- Fill, Hans-Georg, Felix Härer, Fabian Muff, and Simon Curty. 2021. Towards augmented enterprise models as low-code interfaces to digital systems. In *Lecture Notes in Business Information Processing*, vol. 422, 343–352. Cham: Springer International Publishing. ISBN 978-3-030-79975-5. https://doi.org/10.1007/978-3-030-79976-2_22.
- Frank, Ulrich. 2013. Domain-specific modeling languages: requirements analysis and design guidelines. In *Domain Engineering, Product Lines, Languages, and Conceptual Models*, ed. Iris Reinhartz-Berger, Arnon Sturm, Tony Clark, Sholom Cohen, and Bettin Jorn, 133–157. Springer. https://doi.org/10.1007/978-3-642-36654-3_6.
- Frank, Ulrich. 2014. Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges. *Software and Systems Modeling* 13 (3): 941–962. <https://doi.org/10.1007/S10270-012-0273-9>.
- Frank, Ulrich. 2019. Konstruktionsorientierter forschungsansatz. In *Enzyklopädie der Wirtschaftsinformatik: Online-Lexikon*, ed. Norbert Gronau, Jörg Becker, Natalia Kliewer, Jan Marco Leimeister, Sven Overhage. Berlin: GITO. Accessed March 18, 2024. <https://wi-lex.de/index.php/lexikon/uebergreifender-teil/forschung-in-wi/konstruktionsorientierter-forschungsansatz/>.
- Fraser, Martin D., Kuldeep Kumar, and Vijay K. Vaishnavi. 1994. Strategies for incorporating formal specifications in software development. *Communications of the ACM* 37 (10): 74–86. <https://doi.org/10.1145/194313.194399>.
- García-Castro, Raul, Asunción Gómez-Pérez, Óscar Muñoz-García, and Lyndon J.B. Nixon. 2008. Towards a component-based framework for developing semantic web applications. *ASWC Conference*, 197–211. Springer. https://doi.org/10.1007/978-3-540-89704-0_14.

- Getting, Ivan. 1993. Perspective/navigation-the global positioning system. *IEEE Spectrum* 30 (12): 36–38. <https://doi.org/10.1109/6.272176>.
- Goldstein, Herbert. 1980. *Classical Mechanics*. 2nd ed. Upper Saddle River: Pearson. ISBN 9780201029185.
- Götzinger, David, Elena-Teodora Miron, and Franz Staffel. 2016. OMiLAB: an open collaborative environment for modeling method engineering. In *Domain-Specific Conceptual Modeling, Concepts, Methods and Tools*, ed. Dimitris Karagiannis, Heinrich C. Mayr, and John Mylopoulos, 55–76. Springer. https://doi.org/10.1007/978-3-319-39417-6_3.
- Grambow, Gregor, Daniel Hieber, Roy Oberhauser, and Camil Pogolski. 2021. *A Context and Augmented Reality BPMN and BPMS Extension for Industrial Internet of Things Processes*, vol. 436. Lecture Notes in Business Information Processing, 379–390. Cham: Springer. ISBN 978-3-030-94342-4. https://doi.org/10.1007/978-3-030-94343-1_29.
- Grubert, Jens, Tobias Langlotz, Stefanie Zollmann, and Holger Regenbrecht. 2017. Towards pervasive augmented reality: context-awareness in augmented reality. *IEEE Transactions on Visualization and Computer Graphics* 23 (6): 1706–1724. ISSN 1941-0506. <https://doi.org/10.1109/TVCG.2016.2543720>.
- Grum, Marcus, and Norbert Gronau. 2018. *Process Modeling Within Augmented Reality: The Bidirectional Interplay of Two Worlds*. Lecture Notes in Business Information Processing, vol. 319, 98–115. Cham: Springer International Publishing. ISBN 978-3-319-94213-1. https://doi.org/10.1007/978-3-319-94214-8_7.
- Gulden, Jens, and Eric S.K. Yu. 2018. Toward requirements-driven design of visual modeling languages. In *The Practice of Enterprise Modeling - 11th IFIP WG 8.1. Working Conference, PoEM 2018, Vienna, October 31–November 2, 2018, Proceedings*, ed. Robert Andrei Buchmann, Dimitris Karagiannis, and Marite Kirikova, vol. 335. Lecture Notes in Business Information Processing, 21–36. Springer. https://doi.org/10.1007/978-3-030-02302-7_2.
- Härer, Felix, and Hans-Georg Fill. 2020. Past trends and future prospects in conceptual modeling - a bibliometric analysis. In *ER'2020*, 34–47. Springer. https://doi.org/10.1007/978-3-030-62522-1_3.
- Hervás, Ramón, Alberto Garcia-Lillo, and José Bravo. 2011. Mobile augmented reality based on the semantic web applied to ambient assisted living. In *Ambient Assisted Living - Third International Workshop, IWAAAL 2011, Held at IWANN 2011, Torremolinos-Málaga, June 8–10, 2011. Proceedings*, ed. José Bravo, Ramón Hervás, and Vladimir Villarreal, vol. 6693. Lecture Notes in Computer Science, 17–24. Springer. https://doi.org/10.1007/978-3-642-21303-8_3.
- Hevner, Alan R., Salvatore T. March, Jinsoo Park, and Sudha Ram. 2004. Design science in information systems research. *MIS Quarterly* 28 (1): 75–105. <http://misq.org/design-science-in-information-systems-research.html>.
- Hinkle, Dennis E., William Wiersma, and Stephen G. Jurs. 2003. *Applied Statistics for the Behavioral Sciences*, vol. 663. Houghton Mifflin. ISBN 9780618124053.
- Hmidani, O., and E. M. Ismaili Alaoui. 2022. A comprehensive survey of the R-CNN family for object detection. In *2022 5th International Conference on Advanced Communication Technologies and Networking (CommNet)*, 1–6. <https://doi.org/10.1109/CommNet56067.2022.9993862>.
- Holloway, Richard L. 1997. Registration error analysis for augmented reality. *Presence: Teleoperators and Virtual Environments* 6 (4): 413–432. <https://doi.org/10.1162/PRES.1997.6.4.413>.
- Hostettler, Fabian. 2022. Evaluation of a Meta-Modeling Platform for AR and VR Using the Example of ArchiMate. Master's thesis, University of Fribourg. <https://doi.org/10.5281/zenodo.6580270>.
- Hugues, Olivier, Philippe Fuchs, and Olivier Nannipieri. 2011. New augmented reality taxonomy: technologies and features of augmented environment. In *Handbook of Augmented Reality*, ed. Borko Furht, 47–63. Springer. https://doi.org/10.1007/978-1-4614-0064-6_2.
- IKEA. 2023. IKEA Online Shop. Accessed April 28, 2023. <https://www.ikea.com/us/en/p/knarrevik-nightstand-black-30381183/>.

- Jain, Vinod. 2023. *Global Meets Digital: Global Strategy for Digital Businesses - Digital Strategy for Global Businesses*. Productivity Press. ISBN 9781003037446. <https://doi.org/10.4324/9781003037446>.
- Jannaber, Sven, Dennis M. Riehle, Patrick Delfmann, Oliver Thomas, and Jörg Becker. 2017. Designing a framework for the development of domain-specific process modelling languages. In *Designing the Digital Transformation - 12th International Conference, DESRIST 2017, Karlsruhe, May 30–June 1, 2017, Proceedings*, ed. Alexander Maedche, Jan vom Brocke, and Alan R. Hevner, vol. 10243. Lecture Notes in Computer Science, 39–54. Springer. https://doi.org/10.1007/978-3-319-59144-5_3.
- Jiménez, Pablo, Federico Thomas, and Carme Torras. 2001. 3D collision detection: a survey. *Computers & Graphics* 25 (2): 269–285. ISSN 0097-8493. [https://doi.org/10.1016/S0097-8493\(00\)00130-8](https://doi.org/10.1016/S0097-8493(00)00130-8).
- Johannsen, Florian, and Hans-Georg Fill. 2017. Meta modeling for business process improvement. *Business & Information Systems Engineering* 59 (4): 251–275. <https://doi.org/10.1007/S12599-017-0477-1>.
- Jones, Brandon, Manish Goregaokar, and Rik Cabanier. 2023. WebXR Device API. W3C candidate recommendation draft, work in progress, World Wide Web Consortium, Mar 2023. <https://www.w3.org/TR/2024/CRD-webxr-20240210/>.
- Junginger, Stefan, Harald Kühn, Robert Strobl, and Dimitris Karagiannis. 2000. Ein Geschäftsprozessmanagement-Werkzeug der nächsten Generation - ADONIS: Konzeption und Anwendungen. *Wirtschaftsinf.* 42 (5): 392–401. <https://doi.org/10.1007/BF03250755>.
- Kán, Peter, and Hannes Kaufmann. 2012. High-quality reflections, refractions, and caustics in Augmented Reality and their contribution to visual coherence. In *11th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2012, Atlanta, November 5–8, 2012*, 99–108. IEEE Computer Society. <https://doi.org/10.1109/ISMAR.2012.6402546>.
- Karagiannis, Dimitris, and Harald Kühn. 2002. Metamodelling platforms. In *E-Commerce and Web Technologies, Third International Conference, EC-Web 2002, Aix-en-Provence, September 2–6, 2002, Proceedings*, ed. Kurt Bauknecht, A. Min Tjoa, and Gerald Quirchmayr, vol. 2455. Lecture Notes in Computer Science, 182. Springer. https://doi.org/10.1007/3-540-45705-4_19.
- Karagiannis, Dimitris, Heinrich C. Mayr, and John Mylopoulos, eds. 2016. *Domain-Specific Conceptual Modeling, Concepts, Methods and Tools*. Springer. ISBN 978-3-319-39416-9. <https://doi.org/10.1007/978-3-319-39417-6>.
- Karpfinger, Christian. 2022. *Coordinate Transformations*, 603–614. Heidelberg: Springer Berlin Heidelberg, Berlin. ISBN 978-3-662-65458-3. https://doi.org/10.1007/978-3-662-65458-3_53.
- Karsai, Gabor, Holger Krahn, Claas Pinkernell, Bernhard Rumpe, Martin Schneider, and Steven Völkel. 2009. Design guidelines for domain specific languages. In *Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM '09), Orlando*, ed. M. Rossi, J. Sprinkle, J. Gray, and J.-P. Tolvanen, vol. B-108, 7–13. Helsingin Kauppakorkeakoulu. <http://www.dsmforum.org/events/DSM09/Papers/Karsai.pdf>.
- Kaschek, Roland H. 2008. On the evolution of conceptual modeling. In *The Evolution of Conceptual Modeling, 27.04. - 30.04.2008*, ed. Lois M. L. Delcambre, Roland H. Kaschek, and Heinrich C. Mayr, vol. 08181. Dagstuhl Seminar Proceedings. Schloss Dagstuhl - Leibniz-Zentrum für Informatik. <https://doi.org/10.4230/DagSemProc.08181.5>.
- Kasper, Johan, Malin Picha Edwardsson, and Mario Romero. 2017. Occlusion in outdoor augmented reality using geospatial building data. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology, VRST 2017, Gothenburg, November 8–10, 2017*, ed. Morten Fjeld, Marco Fratarcangeli, Daniel Sjölie, Oliver G. Staadt, and Jonas Unger, 30:1–30:10. ACM. <https://doi.org/10.1145/3139131.3139159>.
- Kazhdan, Michael M., Thomas A. Funkhouser, and Szymon Rusinkiewicz. 2003. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *First Eurographics Symposium on Geometry Processing, Aachen, June 23–25, 2003*, ed. Leif Kobbelt, Peter

- Schröder, and Hugues Hoppe, vol. 43. ACM International Conference Proceeding Series, 156–164. Eurographics Association. <https://doi.org/10.2312/SGP/SGP03/156-165>.
- Kekes, John. 1994. The pragmatic idealism of Nicholas Rescher. *Philosophy and Phenomenological Research* 54 (2): 391–394. ISSN 00318205. <http://www.jstor.org/stable/2108498>.
- Kelly, Steven, and Juha-Pekka Tolvanen. 2008. *Domain-Specific Modeling - Enabling Full Code Generation*. Wiley. ISBN 978-0-470-03666-2. <http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470036664.html>.
- Kelly, Steven, Kalle Lyytinen, and Matti Rossi. 1996. MetaEdit+: a fully configurable multi-user and multi-tool CASE and CAME environment. In *Advances Information System Engineering, 8th International Conference, CAiSE'96, Heraklion, Crete, May 20–24, 1996, Proceedings*, ed. Panos Constantopoulos, John Mylopoulos, and Yannis Vassiliou, vol. 1080. Lecture Notes in Computer Science, 1–21. Springer. https://doi.org/10.1007/3-540-61292-0_1.
- Kern, Heiko. 2016. Model interoperability between meta-modeling environments by using M3-level-based bridges. <https://doi.org/10.13140/RG.2.2.29137.33125>.
- Kern, Heiko, Axel Hummel, and Stefan Kühne. 2011. Towards a comparative analysis of meta-metamodels. In *SPLASH'11 Workshops - Compilation Proceedings of the Co-Located Workshops: DSM'11, TMC'11, AGERE'11, AOPES'11, NEAT'11, and VMIL'11, Portland, October 22–27, 2011*, ed. Cristina Videira Lopes, 7–12. ACM. <https://doi.org/10.1145/2095050.2095053>.
- Khan, Nabeel Asif. 2021. Research on various software development lifecycle models. In *Proceedings of the Future Technologies Conference (FTC) 2020*, vol. 3, 357–364. Springer. ISBN 978-3-030-63092-8. https://doi.org/10.1007/978-3-030-63092-8_24.
- Kharroubi, A., R. Billen, and F. Poux. 2020. Marker-less mobile augmented reality application for massive 3D point clouds and semantics. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLIII-B2-2020*: 255–261. ISSN 2194-9034. <https://doi.org/10.5194/isprs-archives-xliii-b2-2020-255-2020>.
- Kim, Dongchul, Seungho Chae, Jonghoon Seo, Yoonsik Yang, and Tack-Don Han. 2017. Realtime plane detection for projection Augmented Reality in an unknown environment. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5985–5989. <https://doi.org/10.1109/ICASSP.2017.7953305>.
- Kitchenham, Barbara. 2004. Procedures for performing systematic reviews. Technical report. https://legacyfileshare.elsevier.com/promis_misc/525444systematicreviewsguide.pdf.
- Krings, Sarah, Enes Yigitbas, Ivan Jovanovikj, Stefan Sauer, and Gregor Engels. 2020. Development framework for context-aware augmented reality applications. In *Proceedings of the 12th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, 1–6, Sophia Antipolis, Jun 2020. ACM. ISBN 978-1-4503-7984-7. <https://doi.org/10.1145/3393672.3398640>.
- Kühne, Thomas. 2006. Matters of (meta-)modeling. *Software and Systems Modeling* 5 (4): 369–385. <https://doi.org/10.1007/S10270-006-0017-9>.
- Landis, J. Richard, and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics* 33 (1): 159. <https://doi.org/10.2307/2529310>.
- Lau, Jey Han, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: automatically evaluating topic coherence and topic model quality. In *Conference of the European Chapter of the Assoc. for Computational Linguistics*, 530–539. ACL. <https://doi.org/10.3115/v1/E14-1056>.
- LaValle, Steven M. 2023. *Virtual Reality*. Cambridge: Cambridge University Press. ISBN 9781108182874. <https://doi.org/10.1017/9781108182874>.
- Lechner, Martin. 2013. ARML 2.0 in the context of existing AR data formats. In *6th Workshop on Software Engineering and Architectures for Realtime Interactive Systems, SEARIS 2013, Orlando, March 17, 2013*, ed. Marc Erich Latoschik, Dirk Reiners, Roland Blach, Pablo A. Figueroa, and Chadwick A. Wingrave, 41–47. IEEE Computer Society. <https://doi.org/10.1109/SEARIS.2013.6798107>.
- Ledeczi, Akos, M. Maroti, A. Bakay, Gabor Karsai, J. Garrett, C. Thomason, G. Nordstrom, J. Sprinkle, and Péter Völgyesi. 2001. The generic modeling environment. In *Workshop*

- on *Intelligent Signal Processing*, Budapest, 17, 01 2001. <https://www.cse.msu.edu/~chengb/CSE891/Techniques/GME/GMEReport.pdf>.
- Li, Menghan, Bin Huang, and Guohui Tian. 2022. A comprehensive survey on 3D face recognition methods. *Engineering Applications of Artificial Intelligence* 110: 104669. ISSN 0952-1976. <https://doi.org/10.1016/j.engappai.2022.104669>.
- Lieberknecht, Sebastian, Andrea Huber, Slobodan Ilic, and Selim Benhimane. 2011. RGB-D camera-based parallel tracking and meshing. In *10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011, Basel, October 26–29, 2011*, 147–155. IEEE Computer Society. <https://doi.org/10.1109/ISMAR.2011.6092380>.
- Likert, Rensis. 1932. A technique for the measurement of attitudes. *Archives of Psychology* 140. [s.n.].
- Mannadiar, Raphaël. 2012. A multi-paradigm modelling approach to the foundations of domain-specific modelling. <https://escholarship.mcgill.ca/concern/theses/kh04dt18n>.
- Maróti, Miklós, Tamás Kecskés, Róbert Kereskényi, Brian Broll, Péter Völgyesi, László Jurácz, Tihamer Levendovszky, and Ákos Lédeczi. 2014. Next generation (meta)modeling: web- and cloud-based collaborative tool infrastructure. In *Proceedings of the 8th Workshop on Multi-Paradigm Modeling co-located with the 17th International Conference on Model Driven Engineering Languages and Systems, MPMMODELS 2014, Valencia, September 30, 2014*, ed. Daniel Balasubramanian, Christophe Jacquet, Pieter Van Gorp, Sahar Kokaly, and Tamás Mészáros, vol. 1237. CEUR Workshop Proceedings, 41–60. CEUR-WS.org. <https://ceur-ws.org/Vol-1237/paper5.pdf>.
- Maxwell, E.A. 1958. *Coordinate Geometry with Vectors and Tensors*. Oxford: Oxford University Press. ISBN 0198531141.
- McCallum, Andrew Kachites. 2002. MALLET: A Machine Learning for Language Toolkit. Accessed July 17, 2023. <http://mallet.cs.umass.edu>.
- Mernik, Marjan, Jan Heering, and Anthony M. Sloane. 2005. When and how to develop domain-specific languages. *ACM Computing Surveys* 37 (4): 316–344. <https://doi.org/10.1145/1118890.1118892>.
- Mertens, Peter, Peter Buxmann, Thomas Hess, Oliver Hinz, Jan Muntermann, and Matthias Schumann. 2023. *Grundzüge der Wirtschaftsinformatik*. Springer Berlin Heidelberg. ISBN 9783662675731. <https://doi.org/10.1007/978-3-662-67573-1>.
- Milgram, Paul, Haruo Takemura, Akira Utsumi, and Fumio Kishino. 1995. Augmented reality: a class of displays on the reality-virtuality continuum. In *Telemanipulator and Telepresence Technologies*, ed. Hari Das, vol. 2351, 282–292. International Society for Optics and Photonics, SPIE. <https://doi.org/10.1117/12.197321>.
- Miller, Joaquin, and Jishnu Mukerji. 2003. MDA Guide Version 1.0. Accessed March 18, 2024. https://www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf.
- Mimno, David M., Hanna M. Wallach, Edmund M. Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27–31 July 2011, John McIntyre Conference Centre, Edinburgh. A meeting of SIGDAT, a Special Interest Group of the ACL*, 262–272. ACL. <https://aclanthology.org/D11-1024/>.
- Mobley, R.K. 2011. *Maintenance Fundamentals*. Plant Engineering, Elsevier Science. <https://doi.org/10.1016/b978-0-7506-7798-1.x5021-3>.
- Moody, Daniel L. 2003. The method evaluation model: a theoretical model for validating information systems design methods. In *Proceedings of the 11th European Conference on Information Systems, ECIS 2003, Naples, 16–21 June 2003*, ed. Claudio U. Ciborra, Riccardo Mercurio, Marco de Marco, Marcello Martinez, and Andrea Carignani, 1327–1336. <http://aisel.aisnet.org/ecis2003/79>.
- Moody, Daniel L. 2009. The physics of notations: toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering* 35 (6): 756–779. <https://doi.org/10.1109/TSE.2009.67>.
- Muff, Fabian. 2020. Interaction Possibilities in VR Conceptual Modeling. Master's thesis, University of Fribourg. <https://doi.org/10.5281/zenodo.10831431>.

- Muff, Fabian. 2024. Appendix A: ARWFMM Use Case Example: FDMM. <https://doi.org/10.5281/zenodo.10639411>.
- Muff, Fabian, and Hans-Georg Fill. 2021a. Initial concepts for augmented and virtual reality-based enterprise modeling. In *Proceedings of the ER Demos and Posters 2021 co-located with 40th International Conference on Conceptual Modeling (ER 2021)*, St. John's, NL, October 18–21, 2021, vol. 2958. CEUR Workshop Proceedings, 49–54. CEUR-WS.org. <https://ceur-ws.org/Vol-2958/paper9.pdf>.
- Muff, Fabian, and Hans-Georg Fill. 2021b. Towards embedding legal visualizations in work practices by using augmented reality. *Jusletter-IT* (27-Mai-2021). ISSN 1664-848X. <https://doi.org/10.38023/e40dcea9-9724-4318-bdca-52ce1cb04e68>.
- Muff, Fabian, and Hans-Georg Fill. 2022a. A framework for context-dependent augmented reality applications using machine learning and ontological reasoning. In *Proceedings of the AAAI 2022 Spring Symposium on Machine Learning and Knowledge Engineering for Hybrid Intelligence (AAAI-MAKE 2022)*, Stanford University, Palo Alto, March 21–23, 2022, ed. Andreas Martin, Knut Hinkelmann, Hans-Georg Fill, Aurla Gerber, Doug Lenat, Reinhard Stolle, and Frank van Harmelen, vol. 3121. CEUR Workshop Proceedings. CEUR-WS.org. <https://ceur-ws.org/Vol-3121/paper11.pdf>.
- Muff, Fabian, and Hans-Georg Fill. 2022b. Use cases for augmented reality applications in enterprise modeling: a morphological analysis. In *Business Modeling and Software Design - 12th International Symposium, BMSD 2022, Fribourg, June 27–29, 2022, Proceedings*, ed. Boris Shishkov, vol. 453. Lecture Notes in Business Information Processing, 230–239. Springer. https://doi.org/10.1007/978-3-031-11510-3_14.
- Muff, Fabian, and Hans-Georg Fill. 2023a. Additional Documents for: “Past Achievements and Future Opportunities in Combining Conceptual Modeling with VR/AR: A Systematic Derivation”, April 2023. <https://doi.org/10.5281/zenodo.7794278>.
- Muff, Fabian, and Hans-Georg Fill. 2023b. ADOxx Library and UseCase Models for the ER23 Publication: A Domain-Specific Visual Modeling Language for Augmented Reality Applications Using WebXR, August 2023. <https://doi.org/10.5281/zenodo.8207639>.
- Muff, Fabian, and Hans-Georg Fill. 2023c. A domain-specific visual modeling language for augmented reality applications using WebXR. In *Conceptual Modeling - 42nd International Conference, ER 2023, Lisbon, November 6–9, 2023, Proceedings*, ed. João Paulo A. Almeida, José Borbinha, Giancarlo Guizzardi, Sebastian Link, and Jelena Zdravkovic, vol. 14320. Lecture Notes in Computer Science, 334–353. Springer. https://doi.org/10.1007/978-3-031-47262-6_18.
- Muff, Fabian, and Hans-Georg Fill. 2023d. Past achievements and future opportunities in combining conceptual modeling with VR/AR: a systematic derivation. In *Business Modeling and Software Design - 13th International Symposium, BMSD 2023, Utrecht, July 3–5, 2023, Proceedings*, ed. Boris Shishkov, vol. 483. Lecture Notes in Business Information Processing, 129–144. Springer. https://doi.org/10.1007/978-3-031-36757-1_8.
- Muff, Fabian, and Hans-Georg Fill. 2024a. Limitations of chatgpt in conceptual modeling: insights from experiments in metamodeling. In *Modellierung 2024 - Workshop Proceedings, Potsdam, March 12–15, 2024*, ed. Holger Giese and Kristina Rosenthal, 8. Gesellschaft für Informatik e.V. <https://doi.org/10.18420/modellierung2024-ws-008>.
- Muff, Fabian, and Hans-Georg Fill. 2024b. M2AR: a web-based modeling environment for the augmented reality workflow modeling language. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24)*, September 2024. ACM. ISBN 979-8-4007-0622-6/24/09. <https://doi.org/10.1145/3652620.3687779>.
- Muff, Fabian, and Hans-Georg Fill. 2024c. Multi-faceted evaluation of modeling languages for augmented reality applications - the case of ARWFML. In *Conceptual Modeling - 43rd International Conference, ER 2024, Pittsburgh, October 28–31, 2024, Proceedings*. Lecture Notes in Computer Science. Springer. https://doi.org/10.1007/978-3-031-75872-0_5.

- Muff, Fabian, Hans-Georg Fill, Eleonora Kahlig, and Wolfgang Kahlig. 2022a. Kontextabhängige Rechtsvisualisierung mit Augmented Reality. *HMD Praxis der Wirtschaftsinformatik* 59 (1): 92–109. ISSN 1436-3011, 2198-2775. <https://doi.org/10.1365/s40702-021-00832-x>.
- Muff, Fabian, Felix Härter, and Hans-Georg Fill. 2022b. Trends in academic and industrial research on business process management - a computational literature analysis. In *55th Hawaii International Conference on System Sciences, HICSS 2022, Virtual Event/Maui, January 4–7, 2022*, 1–10. ScholarSpace. <http://hdl.handle.net/10125/80215>.
- Muff, Fabian, Nathalie Spicher, and Hans-Georg Fill. 2023. Integrating physical, digital, and virtual modeling environments in a collaborative design thinking tool. In *Enterprise, Business-Process and Information Systems Modeling - 24th International Conference, BPMDS 2023, and 28th International Conference, EMMSAD 2023, Zaragoza, June 12–13, 2023, Proceedings*, ed. Han van der Aa, Dominik Bork, Henderik A. Proper, and Rainer Schmidt, vol. 479. Lecture Notes in Business Information Processing, 274–284. Springer. https://doi.org/10.1007/978-3-031-34241-7_19.
- Mütterlein, Joschka. 2018. The three pillars of virtual reality? Investigating the roles of immersion, presence, and interactivity. In *51st Hawaii International Conference on System Sciences, HICSS 2018, Hilton Waikoloa Village, January 3–6, 2018*, ed. Tung Bui, 1–9. ScholarSpace/AIS Electronic Library (AISEL). <https://hdl.handle.net/10125/50061>.
- Mylopoulos, John. 1992. Conceptual modelling and Telos. In *Conceptual Modeling, Databases, and Case: An Integrated View of Information Systems Development*, 49–68. New York: John Wiley & Sons, Inc. ISBN 0471554626.
- National Imagery and Mapping Agency. 1991. Department of Defense World Geodetic System 1984: Its Definition and Relationships with Local Geodetic Systems. Technical Report TR8350.2, National Imagery and Mapping Agency, St. Louis, September 1991. <https://apps.dtic.mil/sti/pdfs/ADA280358.pdf>.
- Newman, David, Arthur U. Asuncion, Padhraic Smyth, and Max Welling. 2009. Distributed algorithms for topic models. *Journal of Machine Learning Research* 10: 1801–1828. <https://doi.org/10.5555/1577069.1755845>.
- Nguyen, Tuong. 2021. 4 Impactful Technologies From the Gartner Emerging Technologies and Trends Impact Radar for 2021. Accessed November 18, 2022. <https://www.gartner.com/smarterwithgartner/4-impactful-technologies-from-the-gartner-emerging-technologies-and-trends-impact-radar-for-2021>.
- Nydegger, Jonas. 2022. Implementation of a Domain-Specific Notation Language for the Meta-Modeling Framework “ModelingToolkitV2”. Master’s thesis, University of Fribourg. <https://doi.org/10.5281/zenodo.7612462>.
- Oberweger, Markus, Paul Wohlhart, and Vincent Lepetit. 2015. Hands deep in deep learning for hand pose estimation. *CoRR*, abs/1502.06807. <http://arxiv.org/abs/1502.06807>.
- Oesterle, Hubert, Jörg Becker, Ulrich Frank, Thomas Hess, Dimitris Karagiannis, Helmut Krcmar, Peter Loos, Peter Mertens, Andreas Oberweis, and Elmar J. Sinz. 2011. Memorandum on design-oriented information systems research. *European Journal of Information Systems* 20 (1): 7–10. ISSN 1476-9344. <https://doi.org/10.1057/ejis.2010.55>.
- OMG. 2012. OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1, April 2012. <https://www.omg.org/spec/UML/ISO/19505-2/PDF>.
- Osterwalder, Alexander, and Yves Pigneur. 2010. *Business Model Generation*. Chichester: John Wiley & Sons. ISBN 9783593394749. https://doi.org/10.1007/978-1-4471-5100-5_13.
- Paquette, Andrew. 2013. *3D Animation*, 239–246. London: Springer London. ISBN 978-1-4471-5100-5. https://doi.org/10.1007/978-1-4471-5100-5_13.
- Peffers, Ken, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. 2008. A design science research methodology for information systems research. *Journal of Management Information Systems* 24 (3): 45–77. <https://doi.org/10.2753/MIS0742-1222240302>.
- Peffers, Ken, Marcus Rothenberger, Tuure Tuunanen, and Reza Vaezi. 2012. Design science research evaluation. In *Design Science Research in Information Systems. Advances in Theory and Practice*, ed. Ken Peffers, Marcus Rothenberger, and Bill Kuechler, 398–410. Berlin,

- Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-29863-9. https://doi.org/10.1007/978-3-642-29863-9_29.
- Petri, Carl, and Wolfgang Reisig. 2008. Petri net. *Scholarpedia* 3 (4): 6477. ISSN 1941-6016. <https://doi.org/10.4249/scholarpedia.6477>.
- Pittl, Benedikt, and Hans-Georg Fill. 2020. A visual modeling approach for the Semantic Web Rule Language. *Semantic Web* 11 (2): 361–389. <https://doi.org/10.3233/SW-180340>.
- Pöhler, Ludger, and Frank Teuteberg. 2021. Closing spatial und motivational gaps: virtual reality in business process improvement. In *European Conference on Information Systems, AIS*. <https://aisel.aisnet.org/ecis2021%5Frp/151>.
- Precht, Peter and Franz-Peter Burkard. 1999. *Metzler Philosophie Lexikon* -. Berlin Heidelberg New York: Springer. ISBN 978-3-476-03780-0. <https://doi.org/10.1007/978-3-476-03780-0>.
- PricewaterhouseCoopers. 2022. US Metaverse Survey: Build a Metaverse Strategy to Deliver Sustainable Business Outcomes. Accessed March 18, 2024. <https://www.pwc.com/us/en/tech-effect/emerging-tech/metaverse-survey.html>.
- Qiang, Jipeng, Ping Chen, Tong Wang, and Xindong Wu. 2017. Topic modeling over short texts by incorporating word embeddings. In *PAKDD*, vol. 10235, 363–374. https://doi.org/10.1007/978-3-319-57529-2_29.
- Quan, Xiaojun, Chunyu Kit, Yong Ge, and Sinno Jialin Pan. 2015. Short and sparse text topic modeling via self-aggregation. In *Joint Conference on Artificial Intelligence*, 2270–2276. AAAI. <http://ijcai.org/Abstract/15/321>.
- Redmon, Joseph, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2016. You only look once: unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, June 27–30, 2016*, 779–788. IEEE Computer Society. <https://doi.org/10.1109/CVPR.2016.91>.
- Riley, K.F., M.P. Hobson, and S.J. Bence. 2006. *Mathematical Methods for Physics and Engineering*. 3rd ed. Cambridge: Cambridge University Press. ISBN 0521679710.
- Ritchey, T. 2006. Problem structuring using computer-aided morphological analysis. *Journal of the Operational Research Society* 57 (7): 792–801. <https://doi.org/10.1057/PALGRAVE.JORS.2602177>.
- Ritzer, G. 2016. *The Blackwell Companion to Globalization*. Blackwell Companions. New York: Wiley. ISBN 9781119250722.
- Roberts, Margaret E., Brandon M. Stewart, Dustin Tingley, Christopher Lucas, Jetson Leder-Luis, Shana Kushner Gadarian, Bethany Albertson, and David G. Rand. 2014. Structural topic models for open-ended survey responses. *American Journal of Political Science* 58 (4): 1064–1082. ISSN 0092-5853, 1540-5907. <https://doi.org/10.1111/ajps.12103>.
- Romero, María Camila, Jorge Villalobos, and Mario Sanchez. 2015. Simulating the business model canvas using system dynamics. In *2015 10th Computing Colombian Conference (10CCC)*, 527–534. <https://doi.org/10.1109/ColumbianCC.2015.7333469>.
- Ropinski, Timo, Steffen Wachenfeld, and Klaus Hinrichs. 2004. Virtual Reflections for Augmented Reality Environments, 311–318. <https://viscom.uni-ulm.de/publications/virtual-reflections-for-augmented-reality-environments/>.
- Roussopoulos, Nick, and Dimitris Karagiannis. 2009. Conceptual modeling: past, present and the continuum of the future. In *Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos*, ed. Alexander Borgida, Vinay K. Chaudhri, Paolo Giorgini, and Eric S.K. Yu, vol. 5600. Lecture Notes in Computer Science, 139–152. Springer. https://doi.org/10.1007/978-3-642-02463-4_9.
- Ruiz-Rube, Iván, Rubén Baena Pérez, José Miguel Mota, and Inmaculada Arnedillo-Sánchez. 2020. Model-driven development of augmented reality-based editors for domain specific languages. *IxD&A*, 18. <https://doi.org/10.55612/s-5002-045-011>.
- Ruminski, Dariusz, and Krzysztof Walczak. 2014. Dynamic composition of interactive AR scenes with the CARL language. In *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications, Chania, Crete, Jul 2014*, 329–334. IEEE. ISBN 978-1-4799-6171-9. <https://doi.org/10.1109/IISA.2014.6878808>.

- Sandkuhl, Kurt, Janis Stirna, Anne Persson, and Matthias WiBotzki. 2014. *Enterprise Modeling - Tackling Business Challenges with the 4EM Method*. The Enterprise Engineering Series. Springer. ISBN 978-3-662-43724-7. <https://doi.org/10.1007/978-3-662-43725-4>.
- Sandkuhl, Kurt, Hans-Georg Fill, Stijn Hoppenbrouwers, John Krogstie, Florian Matthes, Andreas L. Opdahl, Gerhard Schwabe, Ömer Uludag, and Robert Winter. 2018. From expert discipline to common practice: a vision and research agenda for extending the reach of enterprise modeling. *Business & Information Systems Engineering* 60 (1): 69–80. <https://doi.org/10.1007/s12599-017-0516-y>.
- Saxena, Deepak, and Jitendra Kumar Verma. 2022. *Recreating Reality: Classification of Computer-Assisted Environments*, 3–9. Singapore: Springer Singapore. ISBN 978-981-16-7220-0. https://doi.org/10.1007/978-981-16-7220-0_1.
- Schmalstieg, Dieter, and Tobias Höllerer. 2016. *Augmented Reality: Principles and Practice*. Boston: Addison-Wesley. ISBN 9780321883575.
- Seiger, Ronny, Romina Kühn, Mandy Korzetz, and Uwe Aßmann. 2021. HoloFlows: modelling of processes for the Internet of Things in mixed reality. *Software and Systems Modeling* 20 (5): 1465–1489. ISSN 1619-1366, 1619-1374. <https://doi.org/10.1007/s10270-020-00859-6>
- Shahnaz, Farihal, Michael W. Berry, V. Paul Pauca, and Robert J. Plemmons. 2006. Document clustering using nonnegative matrix factorization. *Information Processing and Management* 42 (2): 373–386. <https://doi.org/10.1016/j.ipm.2004.11.005>.
- Siau, Keng, and Matti Rossi. 2011. Evaluation techniques for systems analysis and design modelling methods - a review and comparative analysis. *Information Systems Journal* 21 (3): 249–268. <https://doi.org/10.1111/j.1365-2575.2007.00255.x>.
- Simon, Martin, Karl Amende, Andrea Kraus, Jens Honer, Timo Samann, Hauke Kaulbersch, Stefan Milz, and Horst Michael Gross. 2019. Complexer-YOLO: real-time 3D object detection and tracking on semantic point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2019*. <https://doi.org/10.1109/CVPRW.2019.00158>.
- Singleton, Mark. 2010. *Yoga Body: The Origins of Modern Posture Practice*. Oxford: Oxford University Press. ISBN 9780195395358. <https://doi.org/10.1093/acprof:oso/9780195395358.001.0001>.
- Slater, Mel, and Sylvia Wilbur. 1997. A framework for immersive virtual environments five: speculations on the role of presence in virtual environments. *Presence: Teleoperators and Virtual Environments* 6 (6): 603–616. ISSN 1054-7460. <https://doi.org/10.1162/pres.1997.6.6.603>.
- Sowa, John F. 2000. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove: Brooks/Cole. ISBN 0534949657.
- Stachowiak, Herbert. 1973. *Allgemeine Modelltheorie*. Wien, New York: Springer. ISBN 3211811060.
- Steinberg, David, Frank Budinsky, Marcelo Paternostro, and Ed Merks. 2009. *EMF: Eclipse Modeling Framework 2.0*. 2nd ed. Boston: Addison-Wesley Professional. ISBN 0321331885.
- Stephenson, Reginald J. 1966. Development of vector analysis from quaternions. *American Journal of Physics* 34 (3): 194–201. <https://doi.org/10.1119/1.1972885>.
- Strahinger, Susanne. 1998. Ein sprachbasierter Metamodellbegriff und seine Verallgemeinerung durch das Konzept des Metaisierungsprinzips. In *Modellierung '98, Proceedings des GI-Workshops in Münster, 11.-13. März 1998*, ed. Klaus Pohl, Andy Schürr, and Gottfried Vossen, vol. 9. CEUR Workshop Proceedings. CEUR-WS.org. <https://ceur-ws.org/Vol-9/Strahinger.ps>.
- Sunitha, E.V., and Philip Samuel. 2018. Object constraint language for code generation from activity models. *Information and Software Technology* 103: 92–111. ISSN 0950-5849. <https://doi.org/10.1016/j.infsof.2018.06.010>.
- Sutherland, Ivan E. 1965. The ultimate display. In *Proceedings of the Congress of the International Federation of Information Processing (IFIP)*, vol. 2, 506–508.
- Sutherland, Ivan E. 1968. A head-mounted three dimensional display. In *American Federation of Information Processing Societies: Proceedings of the AFIPS '68 Fall Joint Computer*

- Conference, December 9–11, 1968, San Francisco, - Part I, vol. 33. AFIPS Conference Proceedings, 757–764. Washington, DC: AFIPS/ACM/Thomson Book Company. <https://doi.org/10.1145/1476589.1476686>.
- Syriani, Eugene, Hans Vangheluwe, Raphael Mannadiar, Conner Hansen, Simon Van Mierlo, and Hüseyin Ergin. 2013. AToMPM: a Web-based Modeling Environment. In *Joint Proceedings of MODELS'13 Invited Talks, Demonstration Session, Poster Session, and ACM Student Research Competition co-located with the 16th International Conference on Model Driven Engineering Languages and Systems (MODELS 2013)*, Miami, September 29–October 4, 2013, ed. Yan Liu, Steffen Zschaler, Benoit Baudry, Sudipto Ghosh, Davide Di Ruscio, Ethan K. Jackson, and Manuel Wimmer, vol. 1115. CEUR Workshop Proceedings, 21–25. CEUR-WS.org. <https://ceur-ws.org/Vol-1115/demo4.pdf>.
- Thoring, Katja, Roland M. Müller, and Petra Badke-Schaub. 2020. Workshops as a research method: guidelines for designing and evaluating artifacts through workshops. In *53rd Hawaii International Conference on System Sciences, HICSS 2020, Maui, January 7–10, 2020*, 1–10. ScholarSpace. <https://hdl.handle.net/10125/64362>.
- van Harmelen, Frank, and Annette ten Teije. 2019. A boxology of design patterns for hybrid learning and reasoning systems. *Journal of Web Engineering* 18 (1): 97–124. <https://doi.org/10.13052/jwe1540-9589.18133>. <https://www.cs.vu.nl/~frankh/postscript/JWE2019.pdf>.
- van Krevelen, D.W.F., and Ronald Poelman. 2010. A survey of augmented reality technologies, applications and limitations. *International Journal of Virtual Reality* 9 (2): 1–20. <https://doi.org/10.20870/IJVR.2010.9.2.2767>.
- Vernadat, François. 1996. *Enterprise Modeling and Integration*. Chapman and Hall. ISBN 978-0-412-60550-5. <https://link.springer.com/book/9780412605505>.
- Vernadat, François. 2020. Enterprise modelling: research review and outlook. *Computers in Industry* 122: 103265. ISSN 0166-3615. <https://doi.org/10.1016/j.compind.2020.103265>.
- Vessey, Iris, Venkataraman Ramesh, and Robert L. Glass. 2002. Research in information systems: an empirical study of diversity in the discipline and its journals. *Journal of Management Information Systems* 19 (2): 129–174. <https://doi.org/10.1080/07421222.2002.11045721>.
- Visic, Niksa, Hans-Georg Fill, Robert Andrei Buchmann, and Dimitris Karagiannis. 2015. A domain-specific language for modeling method definition: from requirements to grammar. In *9th IEEE International Conference on Research Challenges in Information Science, RCIS 2015, Athens, May 13–15, 2015*, 286–297. IEEE. <https://doi.org/10.1109/RCIS.2015.7128889>.
- Vogel, Jannis, Julian Schuir, Cosima Koßmann, and Oliver Thomas. 2021. Lets do design thinking virtually: design and evaluation of a virtual reality application for collaborative prototyping. In *European Conference on Information Systems 2021*, 22. https://aisel.aisnet.org/ecis2021_rp/112.
- Wang, X.H., D.Q. Zhang, T. Gu, and H.K. Pung. 2004. Ontology based context modeling and reasoning using OWL. In *IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second*, 18–22. <https://doi.org/10.1109/PERCOMW.2004.1276898>.
- Watson, James L. 2006. *Golden Arches East: McDonald's in East Asia*. Redwood City: Stanford University Press. ISBN 9780804767392.
- Webster, Jane, and Richard T. Watson. 2002. Analyzing the past to prepare for the future: writing a literature review. *MIS Quarterly* 26 (2). <https://www.jstor.org/stable/4132319>.
- Weske, Mathias. 2019. *Business Process Management - Concepts, Languages, Architectures*, 3rd ed. Springer. ISBN 978-3-662-59431-5. <https://doi.org/10.1007/978-3-662-59432-2>.
- West, Stephen, Ross Brown, and Jan Recker. 2010. Collaborative business process modeling using 3D virtual environments. In *AMCIS*, 12. <https://aisel.aisnet.org/amcis2010/249>.
- Weymouth, Stephen. 2023. *Digital Globalization: Politics, Policy, and a Governance Paradox*. Elements in International Relations. Cambridge: Cambridge University Press. <https://doi.org/10.1017/9781108974158>.
- Wieland, Michael, and Hans-Georg Fill. 2020. A domain-specific modeling method for supporting the generation of business plans. In *Modellierung 2020*, 19–21. Februar 2020, Wien,

- Österreich, ed. Dominik Bork, Dimitris Karagiannis, and Heinrich C. Mayr, vol. P-302. LNI, 45–60. Gesellschaft für Informatik e.V. <https://dl.gi.de/handle/20.500.12116/31846>.
- Wieland, Matthias, Holger Schwarz, Uwe Breitenbücher, and Frank Leymann. 2015. Towards situation-aware adaptive workflows: SitOPT - a general purpose situation-aware workflow management system. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom Workshops 2015, St. Louis, March 23–27, 2015*, 32–37. IEEE Computer Society. <https://doi.org/10.1109/PERCOMW.2015.7133989>.
- Wild, Fridolin, Peter Scott, Paul Lefrere, Jaakko Karjalainen, Kaj Helin, Ambjörn Naeve, and Erik Isaksson. 2014. Towards data exchange formats for learning experiences in manufacturing workplaces. In *Proceedings of the 4th Workshop on Awareness and Reflection in Technology-Enhanced Learning, ARTELEC-TEL 2014, Graz, September 16, 2014*, ed. Milos Kravcik, Alexander Mikroyannidis, Viktoria Pammer, Michael Prilla, Thomas Daniel Ullmann, and Fridolin Wild, vol. 1238. CEUR Workshop Proceedings, 23–33. CEUR-WS.org. <http://ceur-ws.org/Vol-1238/paper2.pdf>.
- Wild, Fridolin, Christine Perey, Benedikt Hensen, and Ralf Klamma. 2020. IEEE standard for augmented reality learning experience models. In *IEEE International Conference on Teaching, Assessment, and Learning for Engineering, TALE 2020*, 1–3. IEEE. <https://doi.org/10.1109/TALE48869.2020.9368405>.
- Wilde, Thomas, and Thomas Hess. 2007. Forschungsmethoden der Wirtschaftsinformatik: Eine empirische Untersuchung. *WIRTSCHAFTSINFORMATIK* 49 (4): 280–287. ISSN 0937-6429, 1861-8936. <https://doi.org/10.1007/s11576-007-0064-z>.
- Yin, Kun, Ziqian He, Jianghao Xiong, Junyu Zou, Kun Li, and Shin-Tson Wu. 2021. Virtual reality and augmented reality displays: advances and future perspectives. *Journal of Physics: Photonics* 3 (2): 022010. <https://doi.org/10.1088/2515-7647/abf02e>.
- Zhou, Feng, Henry Been-Lirn Duh, and Mark Billinghurst. 2008. Trends in augmented reality tracking, interaction and display: a review of ten years of ISMAR. In *7th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR 2008, Cambridge, 15–18th September 2008*, 193–202. IEEE Computer Society. <https://doi.org/10.1109/ISMAR.2008.4637362>.
- Zwicky, F. 1989. *Morphologische Forschung: Wesen und Wandel materieller und geistiger struktureller Zusammenhänge*. Schriftenreihe der Fritz-Zwicky-Stiftung. Baeschlin. ISBN 9783855460380.