Data & Knowledge Engineering 165 (2026) 102633

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

ARWFML: A domain-specific modeling language for augmented reality applications☆

Fabian Muff[a], Hans-Georg Fill[b,*]

[a] Independent Researcher, 6390, Engelberg, Switzerland
[b] Research Group Digitalization and Information Systems - Department of Informatics, University of Fribourg, Bd. de Pérolles 90, Fribourg, 1700, Fribourg, Switzerland

ARTICLE INFO

Keywords:
Conceptual modeling
Metamodeling
Augmented reality
Model-driven engineering
Language evaluation

ABSTRACT

Augmented reality (AR) is a technology that embeds digital information onto real-world objects through the use of visual overlays, employing devices such as smartphones, tablets, or head-mounted displays. The goal of AR is to enrich human comprehension and interaction with the physical environment. The creation of AR software applications is complex and traditionally requires advanced coding skills. To simplify AR development, we propose ARWFML, a domain-specific visual modeling language for designing AR scenarios as executable AR applications, enabling users to define augmentations and AR workflows graphically. The research process described in this paper was conducted following the Design Science Research (DSR) paradigm and was organized into four design cycles. First, the modeling language was designed at a conceptual level, followed by the initial implementation of the language on the two-dimensional (2D) metamodeling platform ADOxx. Since 2D modeling tools are inadequate for three-dimensional (3D) modeling, MM-AR, a new web-based 2D and 3D modeling environment for augmented reality applications, was developed in the third design cycle. Additionally, the usefulness of the new modeling environment was demonstrated through use cases of the new modeling language on the platform, and the environment was evaluated across multiple facets, highlighting the potential of the modeling language for AR development. Finally, in the fourth design cycle, the execution of ARWFML models was addressed, proving that ARWFML models are executable as AR applications.

1. Introduction

Augmented reality (AR) plays an important role in the ongoing convergence of the physical and the digital world [1]. At its core, augmented reality enhances the user's perception by superimposing visual overlays such as images, videos, or three-dimensional (3D) visualizations onto physical-world environments in realtime [2,3]. It uses computer vision techniques to align objects in the virtual and physical worlds and displays the virtual information using see-through displays or screens, e.g., on smartphones or head-mounted displays [4]. AR reverts to markers or detectors of real-world objects to determine their location and orientation in three-dimensional space to accurately map visual overlays onto them. For realizing complex AR workflows in practical work scenarios, additional concepts such as the integration of external data sources in combination with triggers, conditions, and actions to process this data become necessary [5].

☆ This article is part of a Special issue entitled: 'Conceptual Modeling and AI_Storey' published in Data & Knowledge Engineering.

* Corresponding author.
E-mail addresses: muff.fabian.research@proton.me (F. Muff), hans-georg.fill@unifr.ch (H.-G. Fill).

https://doi.org/10.1016/j.datak.2026.102633
Received 17 March 2025; Received in revised form 27 May 2026; Accepted 27 July 2026
Available online 4 August 2026

Recent technological advances have made augmented reality affordable via its availability on standard smartphones and tablets [6]. In addition, the future open W3C standard WebXR Device API is being developed for accessing AR devices on the web across a wide variety of hardware form factors [7]. In regard to industrial applications, a market research study conducted by Gartner [8] indicates that AR is a highly promising technology. Additionally, a study by PwC [9] suggests that immersive technology is on the verge of becoming a part of everyday life, providing businesses with easier access to value at scale and making it available for widespread use in industrial scenarios such as maintenance tasks or training [10].

However, creating 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] For easing the creation of AR applications, several proposals have been made in model-driven engineering (MDE) and conceptual modeling. This includes, for example XML and JSON schemas for describing AR scenes in generic, platform-independent formats [11,12] or with a focus on learning experiences [5]; domain-specific languages for creating AR model editors using Vuforia, ARKit, or MRTK [13,14]; or a BPMN extension for representing process information in AR using the Unity platform [10]. In addition, commercial low-code and no-code tools are offered that aim to empower non-technical users to create AR applications. This includes tools such as UniteAR,[5] or Adobe Aero.[6] However, these tools are mostly designed for creating a single AR scene or very simple workflows. Thus, the maturity of existing solutions is rather low. At the same time, the application domain maturity of Augmented Reality in terms of openness and standardization is also considered to be low as the WebXR device API is so far only a candidate recommendation.

From the perspective of design science research, it is thus necessary to invent new solutions, while building partially on existing approaches [15]. In particular, an approach that is capable of representing complex AR workflows for diverse application scenarios, is easily adapted to new requirements, and that is based on open standards could open new opportunities for the field of AR. As we had shown through a previous literature review, such an approach is missing so far [16]. The fields of conceptual modeling and model-driven engineering do however provide the foundations and means to create such adaptable solutions as exemplified by the large number of previous domain-specific modeling languages and tools [17–20]. We therefore pose three research questions following previously elaborated guidelines for design science research [21]:

- RQ1: What are the requirements for a new visual modeling approach for representing complex AR workflows for diverse application scenarios that is based on open standards?
- RQ2: How can we design the new modeling approach compliant with the set of defined requirements?
- RQ3: How can we evaluate the new modeling approach?

This paper is an extension of the previous conference paper [22], including also extended parts of [23,24], which provides further details on the previous design cycles and the evaluation methods. In addition, a fourth design cycle has been added, which led to the implementation of an AR execution engine for interpreting ARWFML models. The paper is now structured according to these four design cycles [25,26] — see Fig. 1. Section 2 details the methodologies used for the development of the proposed approach and illustrates the research process for this contribution. Section 3 describes the fundamental concepts in AR and the most important development platforms for achieving a common understanding. In Section 4, we analyze previous related work in MDE and conceptual modeling in the context of AR. From these insights, we derive generic and specific requirements for a domain-specific visual modeling language for AR applications and present its specification including a first evaluation in Section 5. Section 6 introduces a first implementation of the modeling language on a 2D modeling tool. After the second evaluation, Section 7 addresses the shortcomings of the previous design cycle by introducing a new 3D enhanced modeling platform, including the adapted 3D notation of the *ARWFML*. This includes a third evaluation of the capabilities of the platform and the comprehensibility of the *ARWFML*. Section 8 addresses the execution of *ARWFML* models in an *AR Engine*, followed by a final evaluation through a comparative evaluation of the possible *ARWFML* pipelines. The paper is closed with a conclusion and an outlook in Section 9.

2. Methodology and research process

This research was conducted following the Design Science Research (DSR) paradigm [27,28]. Specifically, the Design Science Research Process Model by Vaishnavi et al. [25,26] was used as an overarching methodology to guide this research. Fig. 1 illustrates the DSR process and the state of the research artifact in terms of the design cycles. The process steps are defined as follows:

(1) **Awareness of the Problem**: This involves the development of an awareness that manifests as an interest in a research gap or an opportunity to improve a solution. This step results in a problem statement and an idea for further research. (2) **Suggestion**: This step envisions a potential solution to the problem in the form of a preliminary design proposal. (3) **Development**: This step implements the preliminary design, resulting in an artifact. (4) **Evaluation**: The previously developed artifact is then subjected to an evaluation appropriate to the type of artifact and its state. The results of the evaluation step are adequate measures, e.g., performance or usability. For early-stage prototypes as we aim for in this paper, the central purpose of our evaluations well be to check to what

[1] https://library.vuforia.com/
[2] https://developer.apple.com/augmented-reality/arkit/
[3] https://developers.google.com/ar
[4] https://github.com/Microsoft/MixedRealityToolkit-Unity
[5] https://www.unitear.com/
[6] https://adobe.com/products/aero.html

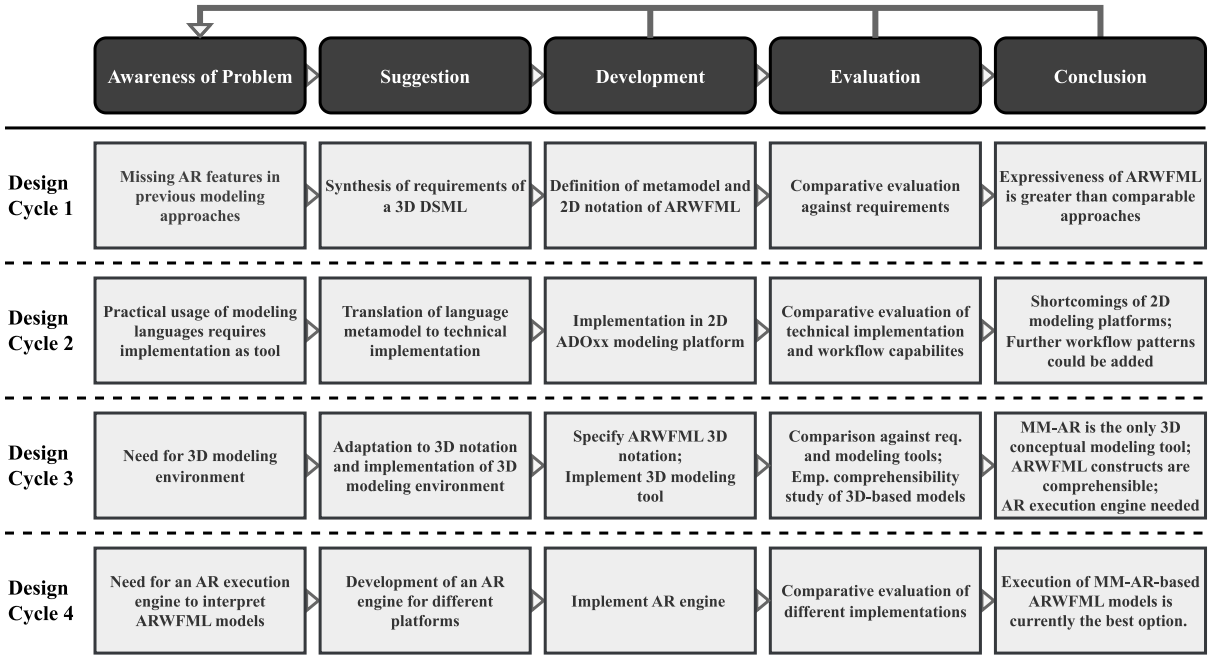


Fig. 1. Design cycles for the continuous development of the ARWFML, which is advanced according to the DSR methodology [27,28] and the Design Science Research Process Model [25,26].

extent the prototypes achieve the intended utility in terms of satisfying the derived requirements [29]. This step always addresses RQ3. (5) **Conclusion**: In this step, the results are analyzed and consolidated. A conclusion is drawn that represents the end of the research effort for this artifact or simply the end of the current research design cycle.

The main research process, which is comprised of four design cycles (DCs), is outlined in the following: (**DC1**) Since previous approaches miss various AR features, we introduce a new modeling language. Several guidelines and methodologies have been proposed for the development of domain-specific modeling languages (DSML), cf. [30–33]. We will mainly follow the macro process proposed by Frank [31], who describes seven phases, including details for each phase. This process will be described in greater detail later in the paper. For an initial assessment, a comparative analysis of features has been conducted with similar languages in the area of augmented reality [34]. (**DC2**) The second cycle includes the implementation of the modeling language on the ADOxx metamodeling platform [35], illustrated through a furniture assembly use case. For the language specification and the creation of the modeling tool we further considered the methodology for agile modeling method development by Visic et al. [33], which focuses on the interplay between a modeling language and algorithms, and the deployment of the modeling tool. Additionally, a comparative evaluation of the technical implementation and its workflow capabilities [36,37] has been conducted. This second evaluation reveals the shortcomings of traditional 2D modeling approaches for three-dimensional AR applications. Consequently, in the third design cycle (**DC3**), we developed a novel 3D-enabled modeling platform, denoted as *MM-AR*. The initial *ARWFML* metamodel has been translated to the *MM-AR* metamodeling language. The notation has been adapted to a 3D format and implemented on the new *MM-AR* modeling platform. The resulting implementation was evaluated in a comparative evaluation against the technical requirements and against other modeling platforms. Furthermore, based on the valid models that can be created with this tool, we conducted a preliminary comprehensibility study to assess the comprehensibility of predefined models and the language constructs of the new modeling language, considering Moody's *Methods Evaluation Model* [38]. (**DC4**) In the fourth design cycle, we focused on the execution aspect of *ARWFML* models, addressing the need for application engines capable of interpreting and running *ARWFML* models as AR application. Specifically, we developed two AR engines for web platforms to ensure broad applicability and platform independence. Furthermore, the evaluation phase involved a comparative evaluation of the different potential *ARWFML* pipelines across platforms.

The following section presents the fundamental topics that are essential for achieving a common understanding of the concepts presented in this paper.

3. Foundations

As augmented reality depends on a range of particular techniques from computer vision to achieve the intended user experience, this section, provides a concise overview of the most significant concepts in order to ensure a unified understanding. Furthermore, we introduce the most important platforms for the implementation of AR applications, as well as the most important web-based modeling platforms for conceptual modeling, which we used for deriving the concepts of the language.

3.1. Augmented reality

Augmented reality is a technology that allows computer-generated virtual objects to be embedded in the real environment [2], thereby creating a three-dimensional alignment between virtual and real objects that allows for interaction in real-time [3].

Augmented reality relies on three core concepts from the field of computer vision [4]: (1) *Detectables/Trackables*, (2) *Coordinate Mappings*, and (3) *Augmentations*. First, for determining the location and orientation of the real-world environment, computer vision algorithms are used to estimate the position and orientation based on 2D or 3D sensor information, e.g., from a camera stream or a LiDAR scanner [39,40]. This detection can either revert to *detectables* in the form of *natural features* or *markers* such as QR codes as surrogates for simplifying the detection and tracking [4]. Coordinate mappings are then needed to align objects in the real and the virtual world to each other. Thereby, 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. Further, *local coordinate systems* are used for any real-world or virtual object. These permit 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*, 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 is 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 [5]. Thereby, triggers include: click, detection, sensor, or timer events; voice commands; entry/exit of defined spatial areas; or, gestures. Conditions specify the branchings into different process flows and actions refer to any change applied to the virtual objects such as the appearance and disappearance of objects or transformations, i.e., rotation, scaling, and positioning.

3.2. AR implementation platforms

For creating AR applications, several development platforms and software development kits (SDK) are provided. Most of them require significant programming skills and are either commercial or closed-source. Examples include the Unity runtime and development environment, Apples ARKit,[7] Wikitude,[8] Vuforia,[9] Kudan,[10] Unreal Engine,[11] or Adobe Aero.[12] In addition, open source platforms and SDKs are available, such as Google ARCore,[13] ARToolKit+,[14] OpenXR,[15] or Holokit.[16]

An alternative to the above platforms and SDKs is the WebXR Device API [7]. It specifies a web Application Programming Interface (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 Chromium-based browsers on the Android operating system,[17] including handheld smartphones and tablets, as well as *head-mounted displays*, e.g., the Microsoft HoloLens 2.[18] Further, WebXR is already included in the WebKit engine used by iOS Safari[19] and is supported by the Apple Vision Pro.[20] However, the Apple implementation currently only allows for VR and not for AR sessions. WebXR does not facilitate the development of technical applications, but applications developed with it are more portable across different platforms.

3.3. Web-based modeling platforms

Today, many browser-based modeling tools exist for both academic and industrial use [41]. These include, for example, established and emerging open source platforms such as WebGME [42], AToMPM [43], or GLSP [44], as well as industrial web-based modeling tools for specific use cases in business and business process modeling, e.g., [45–47] or enterprise architecture, e.g., [48]. Further, the rise of so-called *low-code* and *no-code* platforms [49] has contributed to an increased awareness of the power of visual modeling approaches in software development. Although many of these approaches can be considered as evolutions of model-driven engineering approaches in academia [50], the availability of tools that go beyond the level of academic prototypes and are ready for industrial applications may give additional momentum for further research in this direction [49,51].

---

7 https://developer.apple.com/augmented-reality/arkit/
8 https://www.wikitude.com/
9 https://developer.vuforia.com/
10 https://www.kudan.eu/
11 https://www.unrealengine.com/
12 https://www.adobe.com/products/aero.html
13 https://developers.google.com/ar
14 https://github.com/artoolkitx/artoolkitx
15 https://github.com/KhronosGroup/OpenXR-SDK
16 https://holokit.io/
17 https://caniuse.com/webxr
18 https://microsoft.com/en-us/hololens
19 https://github.com/WebKit/WebKit/tree/main/Source/WebCore/Modules/webxr
20 https://www.apple.com/apple-vision-pro/

Several architecture patterns have been proposed and successfully implemented in the past to realize web-based modeling tools. These include, for example, traditional architectures of web information systems, which differentiate between *front-end and back-end components*. These architectures enable collaborative editing and model persistence, e.g., as recently shown in [44,52]. In addition, approaches exist that only provide browser-based *front-end* components with neither persistency nor collaboration functionalities, e.g., [53]. Furthermore, we can distinguish between web-based tools that target only a *specific set of model/diagram types*, e.g., [53], and such that offer *metamodeling* capabilities, i.e., which permit the definition of arbitrary modeling languages and the generation of according model editors, e.g., [43].

4. Related work

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 *virtual reality / augmented reality* and derived the major research streams in these areas [16]. From the results of this study, we selected the most important contributions in the area of model-driven engineering and conceptual modeling for AR which are related to our approach. These will be briefly characterized in the following.

Ruminski and Walczak [54] describe a text-based declarative language for modeling dynamic, contextual augmented reality environments called *CARL*. They claim that CARL can simplify the creation of AR experiences by allowing developers to create reusable, modular components. Their development approach is based on textual modeling and does not include a visual representation. The created models are meant to be interpreted by an execution engine.

Wild et al. [5] 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). Based on this work, a new IEEE standard for *Augmented Reality Learning Experience Models* (ARLEM) has been developed [55], which includes a reference implementation.[21] ARLEM enables the direct definition of learning workflows within an AR context. However, the textual models for these workflows are stored only at runtime, precluding a definition outside the tool. The *MIRAGE-XR* prototype allows for immersive model creation in three-dimensional space and the runtime interpretation in an AR application.

A similar approach has been developed by Lechner [12]. He proposes the XML-based, textual *Augmented Reality Markup Language* (ARML 2.0) for describing virtual objects, their appearance, and anchors in an AR scene in relation to the real world. ARML 2.0 has been included in a standard issued by the *Open Geospatial Consortium*[22] in the form of an XML grammar.

Ruiz-Rube et al. [13] proposed a text-based model-driven development approach for creating the executable code for 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*.[23] The resulting applications are geared towards modeling itself rather than developing AR applications in general.

Seiger et al. [56] presented *Holoflows*, a modeling approach for creating *Internet of Things* (IoT) processes in augmented reality environments. The approach includes an interface allowing non-experts to design IoT processes without process or modeling knowledge. The approach is specific to the IoT domain and modeling is only possible within the provided AR application.

Grambow et al. [10] introduced a visual 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 to extend business process management software with AR and IIoT capabilities. Further, it supports the modeling of context-aware and AR-enabled business processes. BPMN-CARX extends BPMN with new elements including a graphical notation. The approach is specific to business process modeling and does not seem applicable to other scenarios. The created models can be interpreted in a run-time environment as AR experiences.

Campos-Lopez et al. [14,57] and Brunschwig et al. [58] proposed an automated textual 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, however, there is no possibility to define states or executable workflows. They demonstrate the feasibility of their approach through a prototypical iOS app called *AlteR* that is based on Apple's ARKit.[24] The created models are interpreted to run as a domain-specific language in an AR interface.

Lenk et al. [59] introduced the textual *SSIML* and *X3D* approach, investigated model-driven development and round-trip engineering for 3D web applications by converting text-based models to code for 3D experiences.

In summary, approaches exist for (1) generating specific AR applications based on models and schemata, (2) generating AR-based modeling tools based on MDE, and (3) AR modeling applications based on conceptual modeling languages. However, to the best of our knowledge, there is no visual modeling approach available so far for representing executable AR workflows for diverse application scenarios and that is based on open AR standards. Therefore, we advance in the next section to the definition of the requirements of such a modeling language and its implementation, as well as an exemplary use case.

[21] https://github.com/WEKIT-ECS/MIRAGE-XR
[22] http://docs.opengeospatial.org/is/12-132r4/12-132r4.html
[23] https://github.com/spi-fm/ARE4DSL
[24] https://alter-ar.github.io/

Fig. 2. Seven Phases for Domain-Specific Language Development [31][p.8].

5. Design cycle one: Derivation of the visual modeling language

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 [60]. 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. [17,31]. Related to this is a trend found today in industrial software development with the rise of low-code and no-code approaches which aim at empowering users to develop software with less or no programming expertise [49,50]. As had been mentioned in the introduction and the section on related work, previous similar approaches for creating augmented reality applications either focused on very specific application scenarios, did not permit to specify complex AR workflows or failed to support open standards. We will thus describe in the following a new type of domain-specific visual modeling language for creating augmented reality applications.

In terms of *scope and purpose*, we aim for a language that permits users with no programming expertise to create augmented reality applications that include complex workflows and run in a web browser without further plugins or software components on a broad range of devices, thereby targeting **RQ1**.

5.1. Requirements

Frank distinguishes between *generic* and *specific* requirements that need to be analyzed prior to the language specification [31] — see Fig. 2. As Gulden and Yu pointed out, these requirements have to be carefully balanced for considering trade-offs between different design alternatives [61], especially in terms of simplicity, comprehensibility, and convenience of use of the language [31].

Thus, we defined the following seven **generic requirements** ($\mathbf{GR}_{1-7}$) for our language, which were derived from the suggestions in Frank [31] and in similar fashion by Karsai et al. [30], as well as Jannaber et al. [32] and adapted by the authors to the domain of AR: $\mathbf{GR_1}$: 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 shall 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 shall 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. $\mathbf{GR_4}$: The constructs of the language should allow modeling at a level of detail sufficient for all foreseeable AR applications. $\mathbf{GR_5}$: The language shall provide different levels of abstraction to avoid overloading and thus compromising the proper interpretation of a model. $\mathbf{GR_6}$: There shall be a clear association between the language constructs and the constructs of the relevant target representations in the AR application. $\mathbf{GR_7}$: 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.

Further, we added twelve specific requirements $\mathbf{SR}_{1-12}$ 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 Section 3, (b.) previously identified academic approaches in the area of model-driven engineering for AR [16], and (c.) requirements concerning the implementation of the language in terms of satisfying the purpose of platform-independent execution using WebXR [7]. 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. $\mathbf{SR_1}$: Superimposing virtual objects on the real world (*Augmentation*) is the main functionality of augmented reality applications [5,10,12–14,54,56]. 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. $\mathbf{SR_2}$: To create a realistic AR experience, the digital augmentations superimposed on the physical world must align with the real world [5,13,14]. 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 runtime (*World Origin Reference*). $\mathbf{SR_3}$: 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*) [5,12,14]. $\mathbf{SR_4}$: It must be possible to specify real-world objects that can be tracked during application runtime (*Detectable / Trackable*) [5,10,12–14,54]. 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 runtime, for example using images or 3D object data. $\mathbf{SR_5}$: Specifying the modification of different objects based on different *actions* is a critical functionality of AR applications [5,12,13,54,56]. Thus, the modeling language should permit to define transitions to subsequent actions and to directly manipulate and transform augmentations. $\mathbf{SR_6}$: For realizing complex AR workflows [5,10,56], *triggers and conditions* are required to enable dynamic branchings in AR applications [5,10,13,14,54,56].

The category *Abstraction* refers to a general aspect for creating an AR modeling language and contains only one specific requirement, which details the generic requirement of different abstraction levels ($\mathbf{GR_5}$). $\mathbf{SR_7}$: To reduce complexity and to

separate the different roles required during the specification of AR scenarios, the modeling language shall include concepts for abstraction, e.g., model decomposition, and separation of concerns to allow task sharing among stakeholders with different responsibilities [5,12,13,54]. 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. **SR_8**: Due to the nature of modeling languages, an abstract and a concrete syntax in textual notation needs to be provided [30,31], also for easing future interoperability with previous approaches [5,10,12–14,54,56]. In addition, as visual notations are more intuitive and user-friendly than text-based notations, a two-dimensional graphical notation needs to be specified [10]. Finally, since the AR domain reverts largely to 3D content, specifying models directly in a 3D environment is useful to facilitate spatial imagination [5,14,56]. Thus, a domain-specific modeling language should consider concepts for text-based, 2D visual, and 3D spatial modeling. **SR_9**: To allow for an easy and rapid adaptation of the language as requirements change, the modeling language shall be based on *metamodeling* [13,14,31]. **SR_{10}**: It should be possible to directly feed the model into an AR application for the *execution* of the modeled AR scenario [5,10,13,54]. Thus, a domain-specific modeling language for AR applications shall provide a data format that can be processed by an AR engine during runtime [62] or generate code for creating the AR application itself from the models [10]. **SR_{11}**: 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 large range of devices and enable non-commercial long-term research, the modeling language (*specification*) and code generated from it (*execution*) shall be based on open standards, such as the WebXR Device API [7]. **SR_{12}**: To ensure reproducibility and accessibility, the implementation of the domain-specific modeling language shall be made openly available [5,13,56].

5.2. Language specification

According to Frank the phase of *language specification* contains several parts [31]. The first step is to create a glossary containing all the 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 shall 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 *meta*^{2}-model shall be used. Subsequent to the language specification, Frank 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 language, we first used the metamodeling language of ADOxx [35]. ADOxx was chosen due to its wide usage within projects of the OMiLAB network [63] and the availability of an open platform for the implementation of model editors. The main metamodeling concepts in ADOxx are [35,64]: *ModelType* (Ⓜ), *Class* (Ⓒ), *Relationclass* (Ⓡ), and *Attribute* (Ⓐ). 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 ((Ⓜ), (Ⓒ), (Ⓡ), (Ⓐ)) to indicate the corresponding meta^2-concept.

Fig. 3 shows the metamodel of the new domain-specific modeling language. The modeling language is divided into three separate *ModelTypes* (Ⓜ): *ObjectSpace*, *Statechange*, and *FlowScene*. This results from requirements **GR_2**, **GR_5** and **SR_7**. An *ObjectSpace* (Ⓜ) defines the real world of an AR environment. It contains the two classes *Augmentation* (Ⓒ) and *Detectable* (Ⓒ) as defined by requirements **SR_1** and **SR_4**. Further, augmentations can include other augmentations, indicated by the *child* (Ⓡ) relationclass and they may be connected to Detectables via *anchored* (Ⓡ) relations (**SR_3**). A *Detectable* has an attribute *is_origin* (Ⓐ), specifying if a *Detectable* references the world origin (**SR_2**).

Statechanges are described in the separate ModelType *Statechange* (Ⓜ) - **SR_5** and **SR_7**. Within such models, Augmentations from the *ObjectSpace* model are referenced (*Reference* (Ⓒ)) and changes on their attributes – e.g., a rotation transformation – are expressed via the attribute *statechange_list* (Ⓐ).

The *FlowScene* (Ⓜ) ModelType defines the workflow of the AR application and how it reacts to different environmental conditions (**GR_4**,**SR_6**). Every *FlowScene* contains exactly one *Start* (Ⓒ) and one *End* (Ⓒ) instance (**SR_6**). Each *FlowScene* contains an *ObjectSpace* (Ⓒ) instance, which references an instance of the *ObjectSpace* ModelType. Inside this *ObjectSpace* class instance, the *FlowScene* model defines an *Origin* (Ⓒ), one or multiple *Statechanges* (Ⓒ), *Conditions* (Ⓒ), and *Resolves* (Ⓒ) (**SR_2**,**SR_6**). They are linked to the *ObjectSpace* with the *is_inside* (Ⓡ) 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* (Ⓒ) define requirements which are necessary to trigger the subsequent *Statechanges*, or to trigger *Resolves*, if there are no consecutive *Statechanges* (**SR_6**). Thus, *Statechanges* and *Resolves* are connected to *Conditions* by the *triggers* (Ⓡ) relationclass. *Conditions*, on the other hand, follow an *Origin* or *Statechange* via the *has_condition* (Ⓡ) relationclass. Furthermore, *Conditions* can be associated with an *Observer* (Ⓒ) using the *has_observer* (Ⓡ) relationclass. *Observers* can be used to monitor sensor data or APIs (**SR_6**).

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 1. Thereby, we considered principles from graphical notation design by Moody as far as possible [65]. 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 is planned for the future [66].

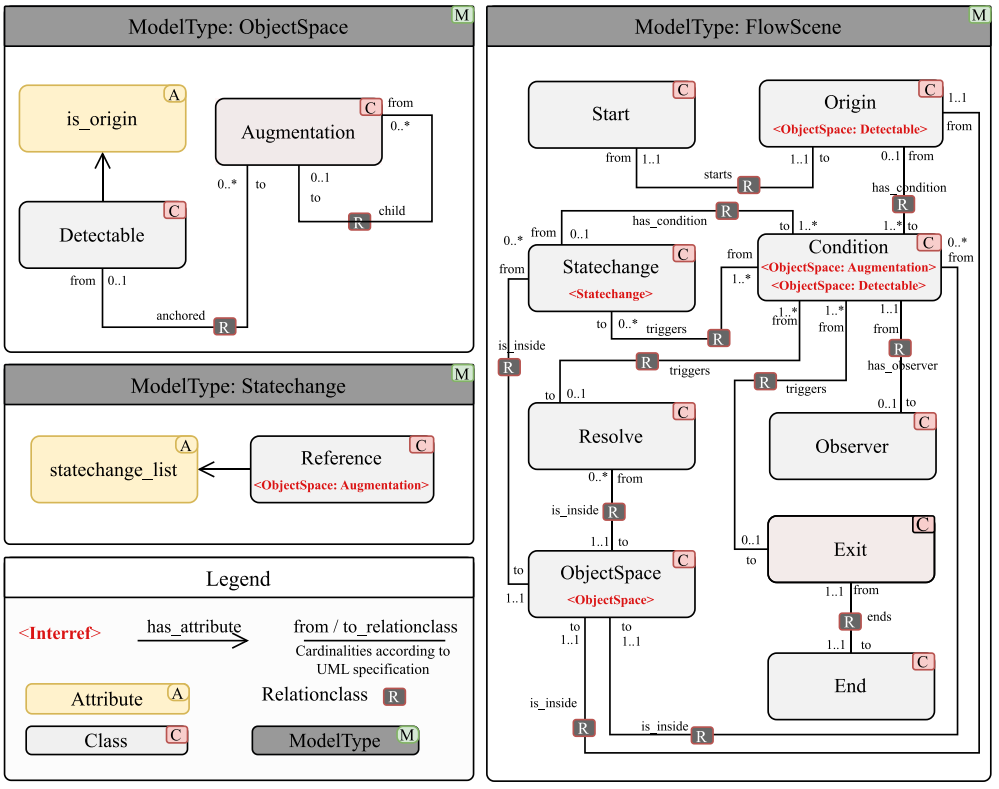


Fig. 3. Metamodel of the DSML for augmented reality applications with the three modeltypes ObjectSpace, Statechange, and FlowScene, as well as a legend.

5.3. Evaluation of design cycle one: Feature comparison

After the language definition of the *ARWFML*, a first evaluation of the language can be conducted. Several techniques can be chosen to evaluate the new modeling language, including feature comparisons, theoretical and conceptual investigations, and empirical evaluations [34]. After the first design cycle, we opted for a feature comparison of the modeling language to previous approaches along the specific requirements that we had formulated.

5.3.1. Feature comparison of the modeling language

The previous approaches we considered for the feature comparison were the ones from Ruminski and Walczak [11], Grambow et al. [10], Seiger et al. [56], Lechner [12], Campos-Lopez et al. [14], Ruiz-Rube et al. [13], and Wild et al. [5].

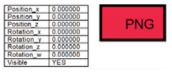
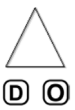
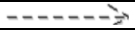
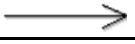
For each specific requirement that we had formulated, we conducted a detailed comparison regarding the dimensions *Domain* and *Abstraction* — see Table 2. The table provide a detailed overview of the features supported by previous approaches and our new modeling language in terms of augmented reality concepts and levels of abstraction. Thereby, we can show that our new modeling language *ARWFML* currently supports 20 out of 25 dimensions of requirements, whereas the next runner-up only supports 17 dimensions.

In regard to *Augmentations* ($\mathbf{SR_1}$), features such as animations, links, checklists, and forms are not yet supported by our language. However, this is more of a technical than a conceptual issue and will be addressed in future versions. The same holds true for area triggers ($\mathbf{SR_6}$). All the other concepts visible in Table 2 are covered by the *ARWFML*.

5.3.2. Summary of evaluation in the first design cycle

Following this initial evaluation, it is possible to conclude the first design cycle. Firstly, *ARWFML* is an expressive domain-specific visual modeling language that is capable of representing complex augmented reality workflows for diverse application scenarios. The modeling language 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 language emphasizes a high level of abstraction and separation of concerns. This abstraction bridges potentially missing knowledge about the technical implementation for AR environments, allowing the user to focus on the content and functionality of AR applications. A first comparison showed, that the *ARWFML* is more expressive than related approaches.

Table 1
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.

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

6. Design cycle two: Implementation in 2D metamodeling tool

When defining new modeling approaches, it is not sufficient to consider only the language requirements, as in design cycle one. It is also necessary to consider how such modeling languages can be technically implemented [31]. Thus, in the second design

Table 2
Feature comparison of the new domain-specific visual modeling language ARWFML against the specific requirements SR_{1-7} of design cycle one. (Y): Requirement met. (N): Requirement not met. (~): Not specified.

	Requirement	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	ARWFML
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
	∑ of supported requirements	7	8	8	12	13	14	17	20

cycle, the metamodel of *ARWFML* has been implemented on a metamodeling tool (**GR_7**). In the following, we will discuss this first implementation, along with a use case demonstration and an evaluation of the technical implementation, as well as the workflow capabilities of the modeling language.

6.1. Implementation on ADOxx

As described in design cycle one, the ADOxx metamodeling language was used to define the *ARWFML* metamodel. Thus, in the second design cycle the modeling language has been implemented using the freely available and open ADOxx metamodeling platform. In this way, we first followed directions from several previous approaches in modeling AR workflows that were also based on existing modeling environments. Although ADOxx does not natively support three dimensional representations, the main focus of this design cycle was to evaluate the consistency and fundamental technical feasibility of the modeling language through an implementation using an established metamodeling platform. Thereby, the modeling language could already be used in first experiments. An alternative would have been to use a purely formal evaluation approach, e.g. based on formalisms such as FDMM [67], however, given the complexity of the language and the goal to create a running implementation, we opted against this. The language files are publicly available via Zenodo [68]. The platform allows the easy definition and adaptation of metamodels based on the ADOxx meta^2-model and the creation of model instances in automatically generated 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**). 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 for enabling the execution of the modeling language (**SR_{10}**). However, the specific part of model execution will be discussed later — see Section 8.

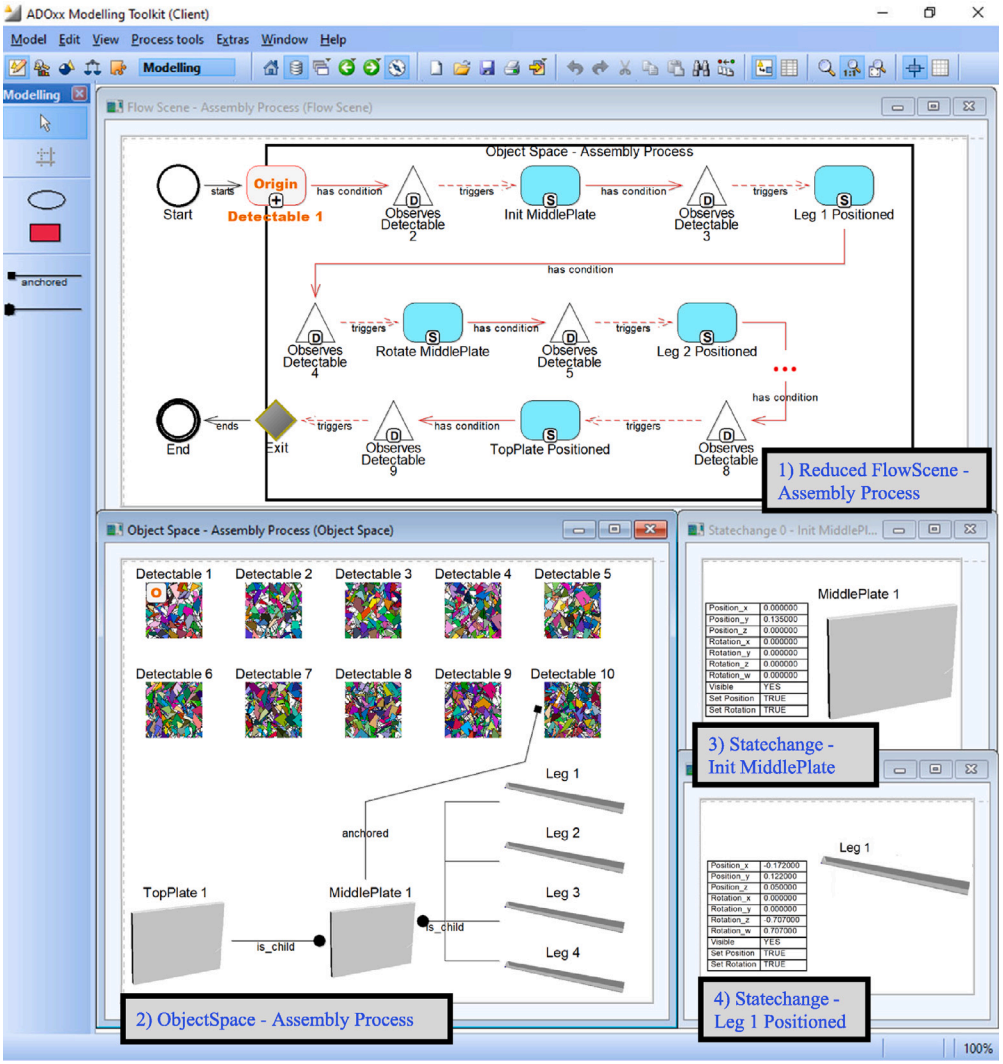


Fig. 4. Screenshot of the ADOxx implementation showing model excerpts for supporting an assembly process in augmented reality: (1) *FlowScene* model of the assembly process. (2) *ObjectSpace* model of the necessary augmentations and detectables using markers. (3) and (4) showing two exemplary *Statechange*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

6.2. ARWFML use case

To demonstrate the use of the modeling language and showcase a practical application, we have developed a use case involving augmented reality-assisted assembly of a bedside table. The goal of this use case is to guide a user through the assembly of a bedside table using an augmented reality application instead of traditional 2D instructions on paper. Fig. 4 shows a screenshot of the implementation in ADOxx. It includes an excerpt of a *FlowScene* model (1), the referenced *ObjectSpace* model (2), and two *Statechange* models (3,4).

In the upper part of Fig. 4, the excerpt of the *FlowScene* model shows how to define the process for assembling the piece of furniture step by step. This includes steps such as turning the pieces into the correct position and attaching them piece by piece. It is important to note that no static flows are defined here but rather *trigger-condition-action* sequences. The *FlowScene* model references one *ObjectSpace* model (2) and several *Statechange* models (3 & 4).

In the lower left part of Fig. 4, the *ObjectSpace* model is shown (2). It includes ten *Detectables* that 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 permit the direct detection of physical objects. Further, the model includes *Augmentation* instances for each part of the

Table 3
Feature comparison of the new domain-specific visual modeling language ARWFML against the specific requirements SR_{8-12} regarding implementation . (Y): Requirement met. (N): Requirement not met. (~): Not specified.

	Requirement	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	ARWFML
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**	**N**
	SR11: Open 3D Standard Support								
	Specification	N	N	N	N	N	N	N	N
	Execution	N	N	N	N	N	N	N	N
	SR12: Openly Available	**N**	**N**	**Y**	**N**	**N**	**Y**	**Y**	**Y**
	∑ of supported requirements	2	3	3	1	3	4	4	4

furniture piece, e.g., "TopPlate 1". These *Augmentations* are provided as GLTF files,[25] which is a common format for 3D objects and their textures. 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. 4, 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 1.

6.3. Evaluation of design cycle two: Technical implementation and workflow capabilities

After the implementation of the *ARWFML* on the *ADOxx* modeling platform [35], a second evaluation of the language can be conducted by evaluating the first technical implementation of the approach in comparison to the requirements and similar approaches — see Section 6.3.1. Furthermore, based on expert feedback received in the first design cycle,[26] we conducted a comparison of the language's workflow capabilities with those of other languages — see Section 6.3.2.

6.3.1. Comparative evaluation of technical implementation

Regarding the technical implementation, two areas can be distinguished. First, as described above, the coverage of the technical requirements of the modeling language – see Table 3, and second, a technical comparison with other approaches – see Table 4.

As for the feature comparison of the non-technical features, the previous approaches considered for the feature comparison of the technical features were the ones from Ruminski and Walczak [11], Grambow et al. [10], Seiger et al. [56], Lechner [12], Campos-Lopez et al. [14], Ruiz-Rube et al. [13], and Wild et al. [5].

As visible in Table 3, the evaluation of the technical requirements of *ARWFML* considers the specific requirements **SR_8-SR_{12}**, i.e., user interaction, metamodeling capabilities, model execution, support for open standards, and availability of according implementations. Thereby, we can show that our new modeling language *ARWFML* currently supports 4 out of 8 requirements.

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 [7] (**SR_{11}**). This would certainly facilitate the specification of models, as 3D modeling greatly facilitates spatial imagination. Furthermore, as mentioned above, model execution (**SR_{10}**) has not been addressed at this point, but will be considered at a later stage (see Section 8). All the other dimensions in Table 3 are covered by the *ARWFML*.

When evaluating modeling approaches for AR scenarios, it is not sufficient to consider only the language requirements for AR concepts, as in the previous section. It is also necessary to consider how such modeling languages can be technically implemented,

[25] https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html
[26] The expert feedback was received in the discussion following the presentation of the approach at the ER conference 2024.

Table 4
Comparison table for related AR modeling approaches. (D): Domain-specific languages. (P): Language profiles. (Y): Dimension covered by approach. (N): Dimension not covered by approach.

	SSIML/ X3D [59]	CARL [54]	BPMN-CARX [10]	ARML [12]	ARE4DSL [13]	ALTER [57]	ARLEM [55]	ARWFML
Language Type								
DSML/Profile	D	D	P	D	D	D	D	D
Model Elicitation Environment								
Textual	Y	Y	Y	Y	Y	Y	Y	Y
Visual 2D	N	N	Y	N	N	N	N	Y
Visual 3D	N	N	N	N	N	N	N	N
Immersive	N	N	N	N	N	N	Y	N
Model Execution								
Code Generation	Y	N	N	N	Y	N	N	N
Model Interpretation	N	Y	Y	Y	N	Y	Y	Y
Resulting Environment								
3D Experience	Y	Y	Y	Y	N	N	Y	Y
Modeling Environment	N	N	N	N	Y	Y	Y	N
Workflow								
Workflow Execution	N	N	Y	N	N	N	Y	Y
Type of Evaluation								
Use Case Demonstration	Y	Y	Y	Y	Y	Y	Y	Y
Implementation	Y	Y	Y	N	Y	Y	Y	Y
Empirical Comprehensibility Study	N	N	N	N	N	N	N	N
Empirical Tool Usability Study	N	N	N	N	Y	Y	N	N
Empirical Simulation	N	N	Y	N	N	N	N	N

including the elicitation and execution of models to create AR scenarios and the necessary technical requirements. The reason for this is that the elicitation of models for 3D space, which is a prerequisite for AR scenarios, is difficult to accomplish in 2D-based modeling environments. In addition, for languages that define workflows, it is important to verify the ability of a language to cover different workflow patterns [37]. Ultimately, the question arises as to what methods can be used to evaluate the modeling approaches in more detail. Thus, for the evaluation part of the second design cycle, in the following, we conduct a comparative evaluation of the technical implementation, including the workflow capabilities, and evaluation types of the major related approaches (cf. Section 4) compared to *ARWFML* — see Table 4. The comparison considers the dimensions *Language Type*, *Model Elicitation Environment*, *Model Execution*, and *Resulting Environment*. Furthermore, the approaches are compared in their capabilities to define workflows *(Workflow)* and applied type of evaluation *(Type of Evaluation)* — see Table 4.

BPMN-CARX [10] is the only language profile. All the other approaches are domain-specific modeling languages. In terms of the model elicitation environment, most approaches [12,13,54,57,59], only support textual model elicitation, with no support for visual 2D, visual 3D, or immersive modeling environments. Only *BPMN-CARX* [10] and the *ARWFML* allow for visual 2D modeling, while *ARLEM* [5] allows for immersive AR modeling. In the dimension of model execution, *SSIML/X3D* [59] and *ARE4DSL* [13] are generating code from the defined models. All the other approaches, including ARWFML, are focusing on model interpretation. Regarding the resulting environment, *ARE4DSL* [13] and *ALTER* [57] are resulting in a modeling environment, while all the other approaches are resulting in 3D experiences. *ARLEM* [5] is the only approach allowing for modeling and executing 3D experiences. Only the three approaches *BPMN-CARX* [10], *ARLEM* [55] and *ARWFML* offer workflow capabilities for specifying complex branching in AR experiences. Regarding the type of evaluation considered by the different approaches, all of them demonstrated use cases. All approaches, except *ARML 2.0* [12], have been evaluated with a first implementation as a proof of concept. No approach evaluated the comprehensibility of the proposed language, and only *ARE4DSL* [13] and *ALTER* [57] have been evaluated regarding the usability of the proposed prototype. In addition, *BPMN-CARX* [10] has been evaluated by conducting an empirical simulation. *ARWFML* has also been implemented and use cases have been shown on the basis of the two-dimensional *ADOxx* metamodeling platform [35] — see Sections 5 and 6.2.

Furthermore, as described above, we got expert feedback in the first design cycle that suggested to further analyze the workflow capabilities of the *ARWFML*. Thus, in the next section, we advance to an evaluation of *ARWFML* against common workflow patterns.

6.3.2. Evaluation of workflow capabilities

Modeling languages for depicting and executing workflows have a long tradition. They permit to structure sequences of activities to complete tasks and may support different types of patterns, which characterize the expressiveness of a language. The most

Table 5
Comparison of support of workflow patterns according to [37] by three AR modeling languages. (Y): Full support. (N): No support. (∼): Partial support.

Workflow pattern	BPMN-CARX [10]	ARLEM [55]	ARWFML
Basic Control Flow Patterns			
Pattern 1 (Sequence)	Y	Y	Y
Pattern 2 (Parallel Split)	Y	N	Y
Pattern 3 (Synchronization)	Y	N	N
Pattern 4 (Exclusive Choice)	Y	Y	N
Pattern 5 (Simple Merge)	Y	Y	Y
Advanced Branching and Synchronization Patterns			
Pattern 6 (Multi - choice)	Y	N	Y
Pattern 7 (Synchronizing Merge)	∼	N	N
Pattern 8 (Multi - merge)	Y	N	Y
Pattern 9 (Discriminator)	∼	N	N
Structural Patterns			
Pattern 10 (Arbitrary Cycles)	Y	Y	Y
Pattern 11 (Implicit Termination)	Y	N	Y
Patterns involving Multiple Instances			
Pattern 12 (Multiple Instances Without Synchronization)	Y	N	N
Pattern 13 (Multiple Instances With a Priori Design Time Knowledge)	Y	N	N
Pattern 14 (Multiple Instances With a Priori Runtime Knowledge)	Y	N	N
Pattern 15 (Multiple Instances Without a Priori Runtime Knowledge)	N	N	N
State-based Patterns			
Pattern 16 (Deferred Choice)	Y	N	N
Pattern 17 (Interleaved Parallel Routing)	∼	N	N
Pattern 18 (Milestone)	N	N	N
Cancellation Patterns			
Pattern 19 (Cancel Activity)	Y	N	N
Pattern 20 (Cancel Case)	Y	N	Y
∑ (fully supported)	**15**	**4**	**8**

significant patterns were summarized by van der Aalst et al. [36,37]. Typical examples for workflow modeling languages include Petri nets [69] or BPMN.[27]

In the following, we compare *ARWFML* against two other AR modeling languages that enable the specification of workflows based on 20 core workflow patterns [37]. Various additional patterns exist. However, van der Aalst et al. [36] mention that these 20 are the most significant patterns. Furthermore, we present a use case for an *ARWFML FlowScene* model that illustrates all the workflow patterns supported by *ARWFML*. Table 5 shows a summary of the modeling languages *BPMN-CARX*, *ARLEM*, and *ARWFML* regarding 20 selected workflow patterns. Thereby, (Y) signifies the support of the pattern, (N) that the pattern is not supported, and (∼) that the pattern is partially supported by the according language.

The analysis reveals significant differences in the workflow capabilities of each modeling language. *BPMN-CARX* shows the most comprehensive support, successfully implementing 15 of the 20 analyzed patterns. This includes comprehensive support for all basic control flow patterns and the majority of advanced branching, synchronization, and multiple instances patterns, indicating a flexible approach to complex workflow scenarios. Since *BPMN-CARX* is an extension of the BPMN modeling language, it comes with the extensive capabilities of the base language and allows for very complex workflows. Conversely, the language is constrained by its focus on process flows in general, with only basic annotations of tasks with AR functionalities. It is not tailored for the definition of spatial models, whereas the other two approaches are explicitly developed for defining workflows for AR.

In contrast, *ARLEM* supports only four patterns. Three basic control flow patterns, and arbitrary cycles. This suggests a focus on simpler, more linear workflows without the need for complex synchronization. *ARLEM*, as well as *ARWFML* do not support patterns involving multiple instances and state-based decisions. Multiple instance patterns describe situations where there are multiple threads of execution active in a process model, each relating to the same activity [37]. Given that there can be only one instance of an AR experience at a given time, these patterns are not considered in *ARLEM*, nor *ARWFML*. The limited support of workflow patterns can restrict the use of *ARLEM* in more complex or variable workflow scenarios. It is important to note that *ARLEM* is designed to define learning experiences and does not claim to allow the definition of highly complex scenarios.

ARWFML supports eight workflow patterns, including the most basic control flow patterns, which are essential for straightforward sequential and parallel processes. This includes the *Sequence*, *Parallel Split*, and *Simple Merge* patterns. However, it shows limitations in handling more complex synchronization and exclusive choice patterns, which are critical for dynamic decision-making scenarios in workflows. In advanced branching and synchronization, *ARWFML* supports patterns like *Multi-choice* and *Multi-merge*, thereby partially outweighing the missing support of Pattern 4 - Exclusive Choice. It lacks support for the *Synchronizing Merge* and

[27] https://www.omg.org/spec/BPMN/2.0/ last visited on: 04.12.2024.

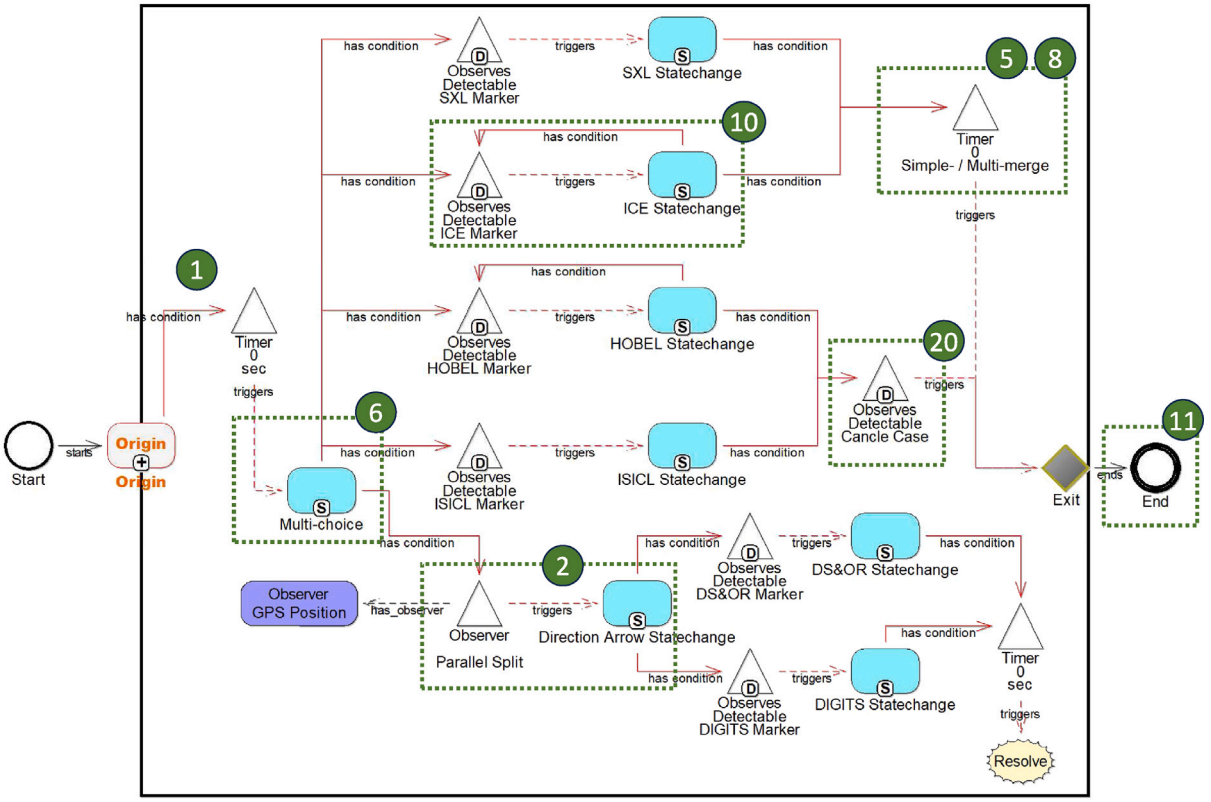


Fig. 5. *ARWFML FlowScene* model illustrating the eight supported workflow patterns by *ARWFML*. Each pattern in the model is marked with a green circle indicating the number of the workflow pattern according to Table 5. (1) *Sequence*; (2) *Parallel Split*; (5) *Simple Merge*; (6) *Multi-choice*; (8) *Multi-merge*; (10) *Arbitrary Cycles*; (11) *Implicit Termination*; (20) *Cancel Case*.

Discriminator patterns, indicating some limitations in splitting and merging multiple concurrent paths within a workflow. The language's support extends to structural patterns, particularly handling arbitrary cycles and implicit terminations effectively. As stated previously, it is not feasible to execute multiple instances of an AR workflow on a single device concurrently, given that there is only one AR session running at a time. Thus, patterns involving multiple instances are not considered. Furthermore, state-based patterns such as *Deferred Choice*, *Interleaved Parallel Routing*, or *Milestone* are not yet supported by *ARWFML*. Cancellation is possible through the *Cancel Case* pattern. Given that *ARWFML* does not have the concept of activities, the *Cancel Activity* pattern is not applicable. In summary it can be stated that ARWFML permits to represent workflows of considerable complexity, which feature for example sequences, advanced branching and merging for OR splits and arbitrary cycles. Although BPMN-CARX supports more workflow patterns, we only deem those related to exclusive choices and synchronization to be the most limiting differences of ARWFML, whereas the other patterns are either not feasible technically or are not fully supported by BPMN-CARX.

To demonstrate the workflow patterns supported by *ARWFML*, Fig. 5 shows an example of a *FlowScene* model. All supported workflow patterns are indicated with the corresponding pattern number in the model. The use case presented in this example depicts a scenario in which an AR user has the potential to visit various research groups on a university campus. At the physical location of each research group on the campus, a marker is placed. As soon as the user comes within proximity to the marker, visual information about the research group is displayed in AR. There are various conditions and paths that can be activated, such as the initiation of a flow, only if the user is within the specified range of *GPS* coordinates. Some areas may be revisited multiple times (*Arbitrary Cycle*), while others may only be visited once.

In summary, *BPMN-CARX* stands out as the most versatile and capable language, especially for environments where complex, multi-faceted workflows are common, but it lacks simplicity since it is based on the BPMN modeling language. *ARLEM* might be more suited for simpler and less dynamic environments. *ARWFML* provides a robust foundation for fundamental and some advanced workflow scenarios. However, further development may be necessary to fully support the more complex and dynamic patterns required in highly variable and decision-intensive workflows. For example, by modifying the *ARWFML* metamodel to allow exclusive choices in the condition concept and introducing the possibility of defining synchronization conditions, patterns 3, 4, 7, and 9 could be supported. Furthermore, by extending the observer concept with incoming messages, pattern 16 could be supported. This would result in the number of supported patterns increasing to 13, thereby approaching the number of patterns that can be supported by the BPMN modeling language.

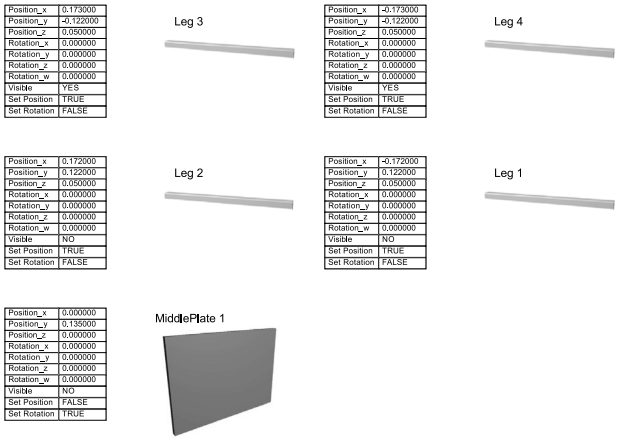


Fig. 6. Example of a Statechange model of the furniture assembly process in the 2D *ADOxx* implementation.

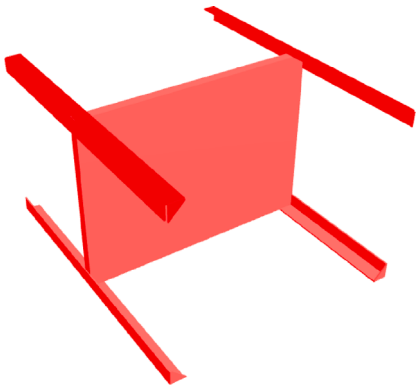


Fig. 7. Example of what a Statechange would look like if it could be modeled in a three-dimensional environment.

6.3.3. Summary of evaluation in the second design cycle

In the evaluation phase of design cycle two, the technical feasibility was demonstrated by implementing the modeling language using the ADOxx metamodeling platform. A technical feature comparison against the initial technical requirements, as well as comparison to the technical implementation of previous approaches indicates that the defined requirements were adequately addressed. In addition, an analysis of the workflow capabilities of *ARWFML* revealed that the language encompasses the most significant workflow patterns, with the potential to encompass a even broader range of patterns.

However, the *ARWFML* approach, after these two design cycles, has some limitations with respect to applicability due to modeling 3D environments in 2D modeling tools. For example, specifying the position of the legs in the application use case described above requires a good understanding of three-dimensional space. It is almost impossible to define position and rotation vectors in 3D space without visualizing them in 3D — see Fig. 6. An example of how this could look like in 3D is visible in Fig. 7. A comparison of Fig. 6 and Fig. 7 reveals a significant discrepancy between the traditional approach and the 3D visualization. While the traditional approach lacks the ability to visualize the result as visible during the execution of the designed AR workflow, the 3D visualization offers a direct representation of the final result during potential execution.

Therefore, in design cycle three (DC3), a new modeling environment has been developed to incorporate the third dimension during visual modeling, enabling 3D modeling in three-dimensional space [70]. This also includes the translation of the *ARWFML* to a three-dimensional notation and the adaptation of the metamodel to the new modeling environment.

7. Design cycle three: A web-based modeling environment for AR applications

This section discusses the third design cycle. This includes the requirements, the conceptual design and the technical implementation of a novel 3D-enabled modeling environment, denoted as *MM-AR*, as well as the *ARWFML* implementation on this environment. The evaluation of design cycle three consists of a comparison against the requirements and comparable modeling tools, as well as a first empirical comprehensibility study.

7.1. Requirements of MM-AR

Based on the *ARWFML* concepts and the general objective of the solution, the following requirements (denoted as **R**) were derived. $\mathbf{R_1}$: In order for the modeling platform to allow for the creation of *ARWFML* models, it must support all defined *ARWFML* concepts. This includes the ability to model with the three different *SceneTypes* on the same platform, interact with and manipulate 3D objects, and dynamically load external images and 3D objects. $\mathbf{R_2}$: As *ARWFML* is a 3D-based language, the modeling environment must allow for easy manipulation of the pose and orientation of 3D objects. $\mathbf{R_3}$: To ensure platform independence and accessibility on various devices, such as smartphones, tablets, and head-mounted displays (HMDs), the environment should be web-based. $\mathbf{R_4}$: The modeling environment must be able to execute the created AR applications on different devices. Thus, to enable execution of models as an AR application, the platform must provide a module for executing the created models as an AR application. $\mathbf{R_5}$: The modeling environment must have the ability to store and access model data in a flexible manner. Thus, modules for exchanging and persisting model data are required. $\mathbf{R_6}$: The AR domain is characterized by a rapidly changing ecosystem, including advancements in tracking technology and evolving legal regulations. To ensure adaptability to new language requirements, the different parts of the modeling environment should be based on a common data structure that allows for flexible adaptation capabilities.

7.2. Implementation of MM-AR

Based on these requirements, a web-based modeling environment has been conceptualized and implemented [71]. This includes (1) a database server with *PostgreSQL*,[28] (2) an *API Server* running as *Node.js*[29] application and *express*,[30] (3) a *Node.js* web server providing (4) the web-page files for a *MM-AR Modeling Client* running *Aurelia2*[31] and the *JavaScript WebGL* visualization framework *THREE.js*.[32] In addition, there is (5) a *Global Shared Datastructure* module defining data templates that can be used in all modules and allows for flexible adaptation capabilities.

For the implementation of all components we used *TypeScript*,[33] which builds on *JavaScript* by adding type checking during development and runs in all major browsers.

7.3. ARWFML metamodel on MM-AR

The new *MM-AR* modeling platform is not just a platform for the modeling of the *ARWFML*, but a fully functional metamodeling platform, cf. [70]. Thus, to implement the *ARWFML* metamodel, the initial metamodel visible in Fig. 3 must be transformed to the *MM-AR* metamodeling language. Fig. 8 shows a simplified version of the transformed metamodel. The new metamodel is very similar to the initial version in Fig. 3. However, the *MM-AR* metamodeling language introduces some additional concepts, e.g., *Ports* and *Roles*. Thus, in the following, we describe briefly the main differences of the translated metamodel (see Fig. 8), in respect to the language specification introduced in Section 5.2.

The *MM-AR* metamodeling language specifies *SceneTypes*, instead of *ModelTypes*. The *classes* in the *ObjectSpace ModelType* and in the *Statechange ModelType* remain the same. In the *MM-AR* metamodeling language, *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 connect to two *roles* denoted as *fr* and *tr*.

The *FlowScene SceneType* contains the same main concepts as the initial metamodel, e.g., *classes* for *Start* and an *End*. However, the *ObjectSpace* class has an *Origin port* that refers to a *Detectable* in the *ObjectSpace* model and an *Exit port*.

7.4. 3D notation

In addition to the changes in the *ARWFML* metamodel, the *MM-AR* environment also requires the adaptation of the *ARWFML* notation in comparison to the 2D implementation introduced in Section 5.2. In contrast to the 2D prototype, the *MM-AR* platform is a 3D-based modeling environment. Furthermore, some visualizations for the *ARWFML* may change during run-time, e.g., *Detectables*, *Augmentations*, or *References*. Thus, the notation of the *ARWFML* has been adapted to meet the new requirements for 3D-based modeling. Table 6 shows the 3D notation of the *ARWFML* language concepts. For each *SceneType*, the visual notation is shown in the *3D Notation* column. If there is a dynamic visualization depending on attribute values, an example visualization is shown in the *Dynamic Example* column. For a semantic description of the different language concepts, see Table 1 in Section 5.2.

7.5. Demonstration

For demonstrating the functionality of our prototype in terms of modeling with the new *MM-AR* environment, we introduce in the following the modeling procedure of *ARWFML*, as well as a use case example.

[28] https://www.postgresql.org/docs/
[29] https://github.com/nodejs/node
[30] https://github.com/expressjs/express
[31] https://github.com/aurelia/aurelia
[32] https://github.com/mrdoob/three.js
[33] https://github.com/microsoft/TypeScript

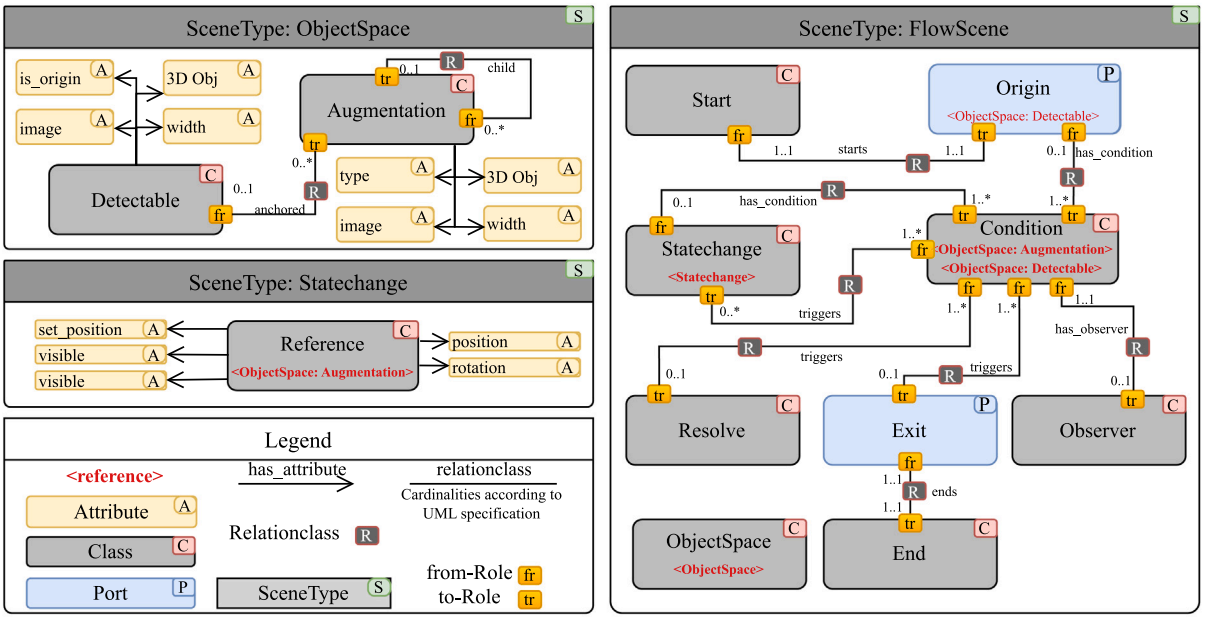
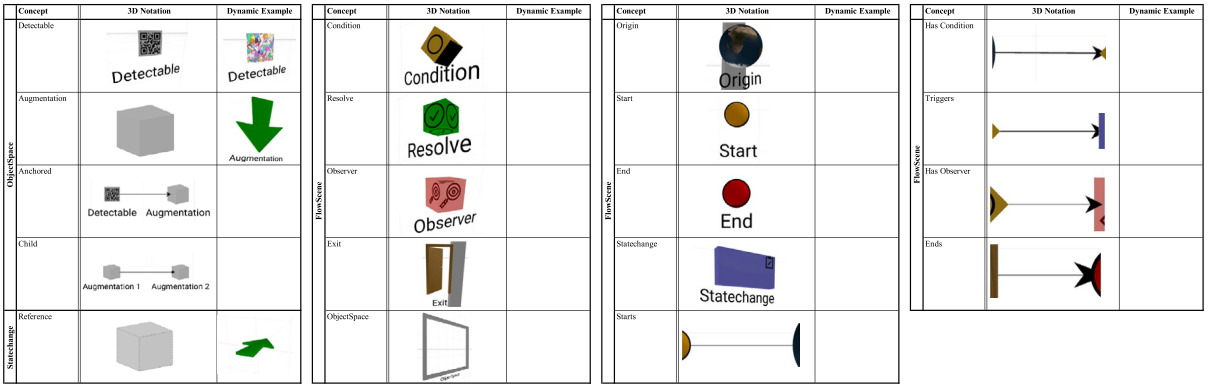


Fig. 8. ARWFML metamodel for ObjectSpace, Statechange, and FlowScene according to the *MM-AR* metamodeling language.

Table 6
3D notation of the *ARWFML*. For each *SceneType*, the visual notation is shown in the *3D Notation* column. If there is a dynamic visualization depending on attribute values, an example visualization is shown in the *Dynamic Example* column.



7.5.1. Add procedure to modeling language and language overview

To ensure a seamless modeling process, in addition to the adapted metamodel and 3D notation, the following modeling procedure [72] for the *ARWFML* is proposed. First, an empty *ObjectSpace* model and a *FlowScene* model must 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 general workflow of the AR application can be created by instantiating empty placeholders for *Conditions* and *Statechanges*. Third, the mapping to the real world must be modeled in the *ObjectSpace*. This means that the origin *Detectable*, as well as all other *Detectables*, must 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 must be created. In each *Statechange* model, *References* to the *Augmentations* to be modified in this *Statechange* must 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.

The following section provides one use case example for modeling with the *MM-AR Modeling Client* to demonstrate the functionality of the *ARWFML* implementation on *MM-AR*.

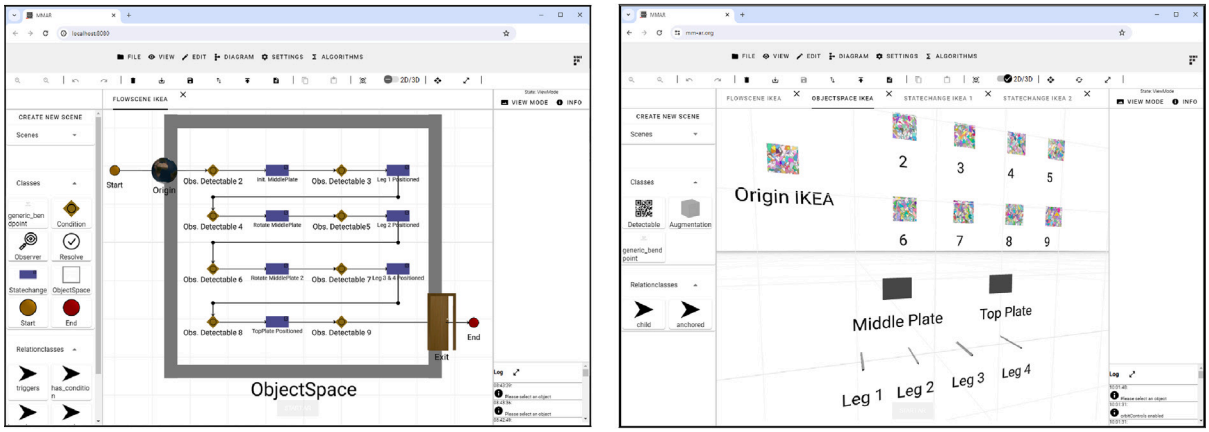


Fig. 9. Screenshots of the FlowScene (left) and the *ObjectSpace* (right) of the IKEA use case in the *MM-AR Modeling Client*.

7.5.2. Furniture assembly use case

The purpose of the second use case is to guide the user through the assembly process of a bedside table in AR. This is the same use case that was used for the first demonstration of the *ARWFML* in the 2D modeling tool *ADOxx* in Section 6.1.

To create these models according to the modeling procedure, it is proposed to start by creating an empty *ObjectSpace*, followed by the creation of the *FlowScene* and the placeholders for the application workflow. The left side of Fig. 9 shows the *FlowScene* model of the use case in the *MM-AR Modeling Client*. 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 of the AR application. In this case, it is a linear workflow with always a *Condition* 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.

The right side of Fig. 9 shows a screenshot of the *MM-AR Modeling Client* while modeling the *ObjectSpace* of the use case in the 3D view of the *MM-AR Modeling Client*. The *ObjectSpace* model contains ten *Detectables*. Every *Detectable* has a unique image as visual representation, coming from the *Image attribute*. 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* of the *Augmentations*. Lastly, for each *Statechange* in the *FlowScene* model, a *Statechange* model is created in the *MM-AR Modeling Client*. Fig. 10 shows two screenshots of the *MM-AR Modeling Client* 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 instances of the *Reference* class 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 *Statechange* references also the "Middle Plate" *Augmentation*, and additionally the "Leg 1" *Augmentation*.

By setting the attributes of the different *References*, e.g., visibility, 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 will visualize the different *Augmentations* when a *Statechange* is triggered. After initially modeling the different models – in this use case an *ObjectSpace*, a *FlowScene*, and seven *Statechanges* – the nine models can be exported as one JSON file through a button in the *MM-AR Modeling Client*. This file can then be interpreted by a potential execution engine.

The *MM-AR* platform has been published as open-source project on GitHub [73] and a live version is accessible online.[34]

7.6. Evaluation of design cycle three: Requirements, features, and empirical study

The DSR methodology allows for many different methods for evaluating an artifact, and there is no rule on when to perform which evaluation [27,28]. One possibility is to check whether the initial requirements are met by the artifact. Another evaluation method is to compare an artifact to previous approaches [34]. Furthermore, it is also possible to conduct empirical studies. Thus, this section evaluates the prototypical implementation of the *MM-AR* environment and its *ARWFML* implementation in three different ways. (1) An evaluation against the initial requirements. (2) A feature comparison of the *MM-AR* environment to other modeling environments, and (3) an empirical comprehensibility study of 3D-based models created with *MM-AR*.

[34] https://mm-ar.org

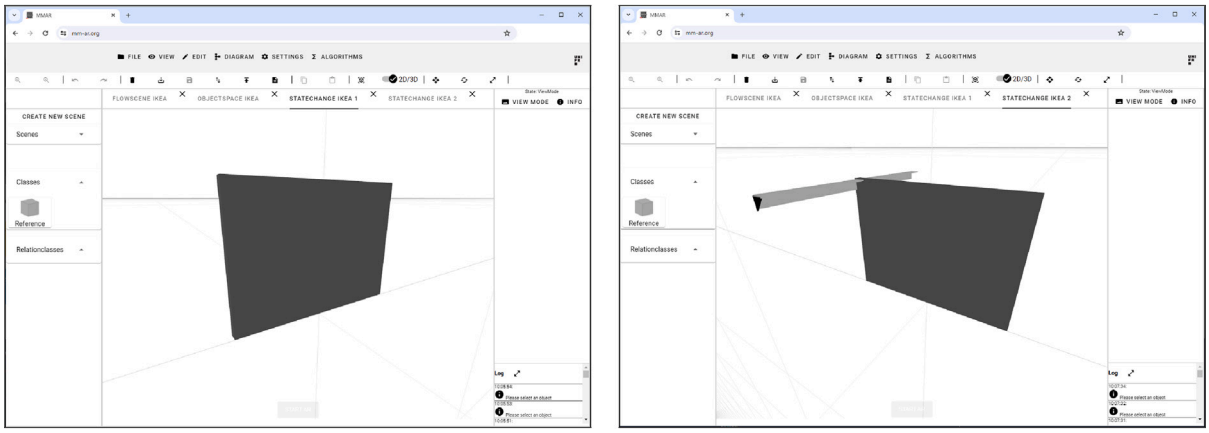


Fig. 10. Screenshots of two Statechanges of the IKEA use case in the *MM-AR Modeling Client* in the 3D view.

7.7. Evaluation against MM-AR requirements

Requirement $\mathbf{R_1}$ states that all *ARWFML* concepts must be supported by the modeling platform. All of the *ARWFML* concepts derived in Section 5 are conceptually supported by the new *MM-AR* environment and most of them are implemented in the modeling language. Thus, the first requirement is met. $\mathbf{R_2}$ demands for the possibility to manipulate 3D objects in a 3D modeling environment. The novel 3D notation for *ARWFML* was implemented in a way in *MM-AR* that facilitates spatial alignment of disparate three-dimensional objects. This is particularly advantageous when modeling *Statechanges*, as they define the manner in which the various *Augmentations* are represented in the real world. Therefore, the second requirement is fully met. $\mathbf{R_3}$ demands for a web-based application. This requirement is fully met by the artifact introduced. $\mathbf{R_4}$ defines the need for a module that is capable of executing the modeled AR applications on different devices. The execution of *ARWFML* models will be addressed in design cycle four. Thus, this requirement is not yet met. $\mathbf{R_5}$ defines the need for flexible data storing capabilities. By allowing for text-based JSON import and export, as well as by introducing an API and a database for data storage, this requirement is met. $\mathbf{R_6}$ defines the need for a common data structure for all modules of *MM-AR*. By introducing the *Global Shared Datastructure* module, this requirement is met. In summary, all but one of the derived requirements ($\mathbf{R_1}$-$\mathbf{R_3}$ and $\mathbf{R_5}$-$\mathbf{R_6}$) are met by the artifact introduced.

7.8. Feature comparison of MM-AR

Furthermore, we evaluated the *MM-AR* environment by comparing its features with those of well-known modeling environments. For this comparison, the modeling environments *ADOxx* [74], *WebGME* [42], *GLSP* [44], and *AToMPM* [43] have been chosen — cf. Section 3.3. For each of these platforms, we compared different dimensions with specific features according to the requirements ($\mathbf{R_1}$-$\mathbf{R_6}$). The dimensions *Modeling Environment*, *Spatial Dimension*, *Persistence*, and *Metamodeling* have been analyzed — see Table 7.

It is evident that all of the analyzed modeling environments are web-based, except for the *ADOxx* modeling platform. Additionally, *MM-AR* is the only environment with native *WebXR* support. Regarding *Spatial Dimensions*, *MM-AR* is the only platform that allows for 3D modeling and interaction in 3D space. While all platforms support file import/export, only *GLSP* and *MM-AR* can be run with or without an underlying database. All of the analyzed platforms are metamodeling platforms that allow for the definition and adaptation of modeling languages. In summary, it can be demonstrated that *MM-AR* supports all nine features, while the next runner-up only supports six. Unlike *MM-AR*, none of the other options allowed for 3D modeling as required by the *ARWFML*.

Having evaluated the *MM-AR Modeling Platform* against its requirements and against other modeling tools, the next logical step would be to evaluate the language and platform with users. When considering the involvement of users, recent related approaches conducted usability [13,57], or simulation studies [10]. Such studies are strongly dependent on software tools. As a result, they do not evaluate the modeling language itself, but rather a combination of the modeling language and the implementation in the tool. However, *MM-AR* is a prototypical modeling environment that has not yet reached a level of maturity sufficient to empirically evaluate its usability due to the high expectations users have for such tools today. Nevertheless, it allows expert modelers to create valid *ARWFML* models that could then be interpreted by an *AR Engine* that takes models from the *MM-AR Modeling Client* as input to execute modeled AR applications. Thus, the evaluation of the comprehensibility of the modeling language should be first conducted independently of the software tool, e.g., as proposed by Bork et al. [75]. Consequently, models created with *MM-AR* are well-suited for an empirical evaluation of the comprehensibility of the modeling language.

7.9. Empirical user study for ARWFML model comprehensibility

This section presents the design and findings of an empirical user study conducted to assess the comprehensibility of *ARWFML* in 3D notation. We describe the study design, followed by data on the experimental subjects, the evaluation metrics used, the study results, and a discussion of potential threats to the study's internal and external validity.

Table 7
Feature comparison list of the new MM-AR modeling environment against different 2D modeling platforms. (Y): Feature supported. (N) Feature not supported.

Dimension/Platform	ADOxx [74]	WebGME [42]	GLSP [44]	AToMPM [43]	MM-AR [73]
Modeling Environment					
Web-based Modeling	N	Y	Y	Y	Y
Native WebXR Support	N	N	N	N	Y
Spatial Dimension					
Modeling of 2D Models	Y	Y	Y	Y	Y
Modeling of 3D Models	N	N	N	N	Y
Interaction in 3D Space	N	N	N	N	Y
Persistence					
File Import/Export	Y	Y	Y	Y	Y
Database Storage	Y	Y	Y	N	Y
Functional without Database	N	N	Y	Y	Y
Metamodeling					
Definition of Metamodels	Y	Y	Y	Y	Y
Total	4	5	6	5	9

Table 8
Categories of survey question (Q1 - Q5) for evaluating *ARWFML* model understanding.

Q	Use case	Purpose
Q1	Color Brick	Prediction of outcome showing different Statechanges.
Q2	Color Brick	Prediction of outcome when differing Conditions.
Q3	Solar System	Prediction of outcome showing different Statechanges.
Q4	Solar System	Prediction of outcome with different Statechanges and Conditions.
Q5	Industrial Machine	Prediction of outcome showing different Statechanges.

7.9.1. Study design

To evaluate the comprehensibility of *ARWFML*, our study employs a between-subject design [76]. Thereby, two user groups were independently tested. This design was chosen to identify differences between user groups with different backgrounds. The main threats to the validity of this setup will be addressed later. The experiments were conducted at the end of 2023.

Each study consisted of two parts: (1) an introduction to the *Augmented Reality Workflow Modeling Language*, which covered the basic concepts of AR, the specific *ARWFML* language concepts, and an introduction to the new *MM-AR* environment featuring the 3D notation. The introduction included a 45-minute presentation with examples of the different language concepts of *ARWFML*. Following the presentation, participants were asked to complete a questionnaire regarding their demographics, understanding of *ARWFML* models, and general comprehension of the different *ARWFML* language concepts. To assess model understanding, participants were shown five sections of screenshots from *ARWFML* models created with *MM-AR*, each with three different possibilities for the resulting workflow in the resulting AR application. The participants were then asked to choose the correct result for each question.

The purpose of the questions was to assess participants' comprehension of the visual models and determine the predictability of the model outcomes. Table 8 shows the categories of questions related to model understanding, including the use case represented and the purpose of each category.[35] A total of three use cases were included. The first use case guides potential users through the assembly of different colored bricks. The second use case, the solar system use case, provides a learning opportunity by displaying the different planets and moons in our solar system. The third use case, the industrial machine use case, guides users through a working process, indicating which buttons to press. An example question on model understanding in the color brick use case is shown in Fig. 11.

Additionally, participants were presented with statements about the *ARWFML* modeling language and asked to identify all correct statements in five separate questions (see general questions GQ1-GQ5 in Table 9). The purpose of the general questions was to gather information about the specific concepts of the modeling language. GQ1 had the goal of determining if participants could recall the different *ARWFML SceneTypes* presented in a selection of unrelated other *SceneTypes*. GQ2 aimed to determine whether participants were able to recall the purpose of the three different *SceneTypes* by selecting true statements about each one. Questions GQ3 to GQ5 were presented separately for each *SceneType* of *ARWFML*. The participants were presented with different choices of different concepts. They had to select only the concepts that are part of the *SceneType* to which the question refers. The purpose of these questions was to evaluate the participant's understanding of the *SceneTypes* of the *ARWFML*.

The sections that follow will cover the experimental subjects, evaluation metrics, and questionnaire results in quantitative terms.

[35] The survey is available on: https://doi.org/10.5281/zenodo.13219036.

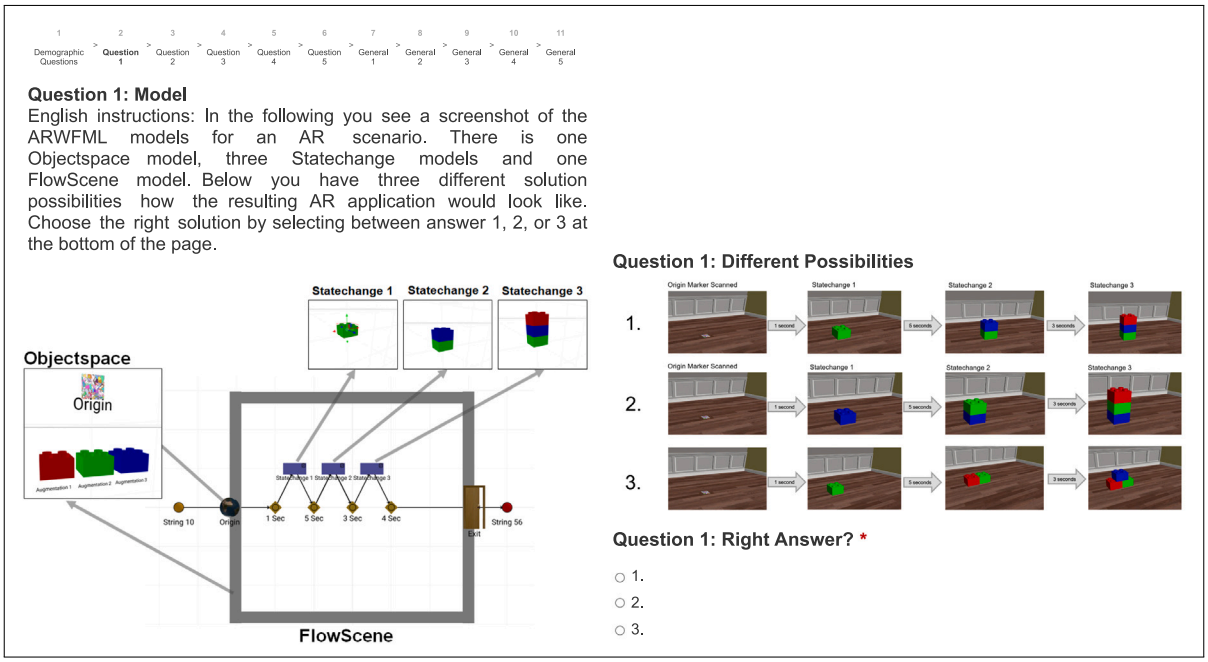


Fig. 11. Examples of the first survey question in the model understanding part of the study.

Table 9
General questions (GQ1 - GQ5) for evaluating *ARWFML* language understanding.

GQ	Question	Purpose
GQ1	Which are the 3 different types of models that can be created with ARWFML?	SceneTypes
GQ2	ARWFML defines the three types ObjectSpace, Statechange, and FlowScene. Choose the three right statements.	SceneType Purpose
GQ3	Which concepts are available in an ObjectSpace model? Choose the two 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 seven introduced modeling concepts of a FlowScene model.	FlowScene Understanding

7.9.2. Experimental subjects

To ensure that the subjects had at least a basic understanding of modeling, we recruited them from university courses in "Introduction to Business Informatics" and "Databases" at the University of Fribourg, where various modeling languages are taught. Our sample consisted of 35 students, aged between 18 and 44 years, the majority of whom were under 24 years of age. The study involved participants with varying levels of education, including high school graduates (25), individuals with Bachelor's degrees (7), Master's degrees (2), and Doctoral degrees (1).

To evaluate the participants' familiarity with *conceptual modeling* and *augmented reality*, we gathered data using five-point Likert scales [77]. Responses are categorized as "Very Familiar", "Familiar", "Somewhat familiar", "Somewhat unfamiliar", and "Very unfamiliar", with corresponding participant counts for each category. The data indicate a range of exposure and understanding among the participants in these two domains. Of the 35 respondents, 18 rated their familiarity with *conceptual modeling* as "Somewhat familiar" or higher, and four considered themselves "Familiar". In the context of *augmented reality*, 25 participants stated that they are "Somewhat unfamiliar" or "Very unfamiliar".

7.9.3. Evaluation metrics

Measuring the comprehensibility of a modeling language is difficult, given that modeling is typically performed using a modeling tool, making it challenging to separate the language from the tool for evaluation. Moody's *Methods Evaluation Model* [38] defines four measures to be considered when evaluating information systems design methods in general: *Performance*, *Perceptions*, *Intentions*, and *Behavior*. Since we wanted to objectively evaluate only the modeling language that is not yet used in practice, we only evaluated *Performance*, which is composed by the measures *Actual Efficiency* and *Actual Effectiveness* [78]. 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 reaches its

Table 10
Percentage of correct responses across studies for questions on model understanding (Q1 to Q5).

	Study 1	Study 2	Total
Q1	0.94	0.95	0.94
Q2	0.88	0.95	0.91
Q3	0.81	0.84	0.83
Q4	0.94	0.95	0.94
Q5	1.00	0.95	0.97

Table 11
Average correct answer rate for questions on *ARWFML* language understanding (GQ1 to GQ5).

	Study 1	Study 2	Total
GQ1	0.91	0.88	0.89
GQ2	0.78	0.87	0.83
GQ3	0.90	0.92	0.91
GQ4	0.92	0.86	0.89
GQ5	0.84	0.79	0.81

objectives. To evaluate the performance of an unproductive language, we have simplified the assessment to determine whether participants can distinguish between different model scenarios and understand the different language concepts. This is achieved by presenting them with various models for validation and asking general questions about the language (see Table 9).

7.9.4. Results

This section presents the results of the empirical comprehensibility study. The results for Q1-Q5 are discussed first, followed by the results of GQ1-GQ5. Additionally, a correlation analysis for Q1-Q5 and GQ1-GQ5 in regard to familiarity with conceptual modeling and augmented reality is presented.

Q1-Q5: Table 10 shows the percentage of correct answers for each of the five questions on *ARWFML* model understanding (Q1 to Q5) in the two studies. The data is presented to show the percentage of participants choosing the right answer out of three given possibilities, e.g., as visible in Fig. 11. A "Total" row aggregates the performance of both studies, providing an overall success rate for each question.

The results show a high overall percentage of correct responses for Q1 (0.94), Q4 (0.94), and Q5 (0.97) on average, indicating a robust understanding of the models. Q2 had moderate accuracy (0.91) with a notable increase in study group two. Q3 had the lowest average accuracy (0.83), which is still moderate. These findings indicate that the participants understand the models presented and the intended AR application scenario derived from these models.

GQ1-GQ5: Table 11 shows the average accuracy of responses across the two studies for the five general question on *ARWFML* language understanding (GQ1- GQ5) — refer to Table 9. The questions were formatted as multiple-choice, with different amounts of possible answers and correct responses. The bottom row displays the total average value for correct answers over all study participants. These five questions provide insights into the relative difficulty or clarity of the different concepts of the modeling language (see *Purpose* column in Table 9).

The results of both studies indicated that question GQ3 (*ObjectSpace* Understanding) consistently achieved the highest and second-highest average scores for correct responses (scores of 0.90 and 0.92), suggesting that it was possibly the simplest or most comprehensible language concept. The lowest average correct answer rate in the first study was observed for GQ2 (SceneType Purpose) at 0.78, while in study two it was observed for GQ5 (*FlowScene* Understanding) at 0.79. This suggests that these concepts were the most difficult or least clear. In the second group, the mean correct answer rates for questions GQ2 and GQ3 were generally higher than in the first group, but lower for questions GQ1, GQ4, and GQ5. The total average correct answer rate across both groups for each question shows that, overall, GQ3 (Object Space model) had the highest average correct answer rate (0.91), while GQ2 (SceneType Purpose) had the lowest (0.83). This indicates that the *ObjectSpace* concept was the best understood or easiest language concept (GQ2), while it is more challenging for participants to distinguish the general purpose of the different *SceneTypes* (GQ3).

Correlation: To investigate the performance disparities related to varying degrees of familiarity with conceptual modeling or AR, we calculated the Pearson correlation coefficients [79] between the measures of familiarity and the rates of correct answers for various *ARWFML* questions, based on a five-point Likert scale. Age and education level were not included in the correlation analysis because most of the participants fell into the same categories.

The correlation coefficients were all in the range between −0.12 and 0.39 (see Table 12). According to [80], value in the range −0.3 to 0.3 are classified as "negligible correlation". Four coefficients marginally exceeded this range, yet they are still categorized under "low correlation". A t-test was performed to assess the statistical significance of the observed results. The analysis revealed that none of the values tested exhibited a significant correlation at the significance level $\alpha = 0.01$. However, GQ1 was the only question that demonstrated a significant correlation between familiarity with conceptual modeling and the correct answer rate at the significance level $\alpha = 0.05$. It is important to highlight that the study involved only 35 participants, which could affect the robustness and generalizability of the correlation analysis results.

Table 12
Correlation coefficients between correct answer rates for *ARWFML* questions and familiarity measures for conceptual modeling or augmented reality. (*) Significant correlation at $\alpha = 0.05$.

	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

These findings provide insight on aspects of the *ARWFML* language that could benefit from more detailed explanations or improved educational resources. The perceived level of difficulty or ease by the participants in responding to the questions highlights the areas requiring improvement. In summary, the language concepts and models created with *ARWFML* are highly comprehensible. For model understanding, responses were accurate in more than 80% of instances, with an average correctness exceeding 80% across eight of ten domains, and surpassing 90% in four domains regarding general language comprehension. There were no discernible differences in the number of correct answers considering different levels of familiarity with augmented reality or conceptual modeling. This suggests that deep knowledge in these areas is not necessary for understanding *ARWFML* models created on the basis of the improved 3D notation.

7.9.5. Threats to validity

To assess the comprehensibility of *ARWFML*, a controlled experiment was conducted with the objective of ensuring both internal and external validity. In order to ensure external validity, participants with at least minimal modeling knowledge were recruited. However, the university course setting and predefined use cases may limit the generalizability of the results to real-world contexts. Moreover, the evaluation of the language through the use of screenshots rather than hands-on interaction with *MM-AR* may result in a limitation of the level of comprehension that could be obtained through active engagement. Furthermore, the emphasis on simple scenarios raises concerns about the applicability of the language to complex situations. We explicitly acknowledge the limitation of using only screenshots rather than the actual modeling platform and the use of students rather than practitioners in augmented reality as experimental subjects. This is due to the fact that the developed prototype does in our view not yet reach maturity level necessary for productive software applications, which would be required for sound user studies.

With regard to internal validity, the use of screenshots to test model understanding may not fully measure *ARWFML*'s performance. Direct interaction with models could provide authenticity, but also introduce biases in terms of comprehensibility. Furthermore, participant selection may introduce bias due to varying proficiency levels. Finally, sequencing the prototype introduction before administering the questionnaire could influence perceptions, but the interdependence between the language and the application may mitigate this.

7.9.6. Summary of evaluation in the third design cycle

After the evaluation phase of the third design cycle (DC3), we can conclude that *MM-AR*, in comparison to other modeling tools, is the only metamodeling platform that supports 3D conceptual modeling. One of the key insights that can be derived is that the empirical evaluation of such languages requires the availability of adequate tool support. This principle also applies to the assessment of a language's comprehensibility. The reason is that modeling languages for augmented reality typically require the specification of spatial information in three-dimensional space, which cannot be accomplished in 2D modeling environments. The introduction of the *MM-AR* modeling platform allows for the creation of formally correct models that were subsequently utilized as input for an empirical user study. In summary, the study showed that the newly introduced 3D notation of *ARWFML* and its associated language concepts are in general highly comprehensible for users with various levels of knowledge in augmented reality and conceptual modeling.

Given the existence of an approach that enables the creation of valid *ARWFML* models, it is obvious to inquire as to the manner in which these *ARWFML* models can be ultimately executed as AR applications. We have discussed the requirements to execute the created *ARWFML* models as AR application during the specification of the modeling language — cf. $\mathbf{SR_{10}}$ in Sections 5.2 and 6.3.1, as well as **R4** in Section 7.1. However, up to this point, this has not been considered. Thus, in the following section, we introduce different approaches to execute *ARWFML* models as AR experience.

8. Design cycle four: AR execution engine

For the demonstration of the execution of the generated models, an *AR Engine* has been developed. In the initial implementation (see Section 8.1), the WebXR device API [7] was selected as the development platform due to its advantageous status as an open-source standard available to everyone. Moreover, the utilization of open 3D standards is a prerequisite, as specified in Section 5.1 ($\mathbf{SR_{11}}$ and $\mathbf{SR_{12}}$). The second implementation (see Section 8.2) is built with the same technology. However, it is not based on *ADOxx*, but on the new *MM-AR* platform.

8.1. AR engine implementation based on ADOxx

The initial implementation (v1.0) of the *AR Engine* is based on the *ADOxx* implementation of *ARWFML* (see Section 5), with the objective of transforming *ADOxx ARWFML* models into AR applications. For this purpose, a software component has been designed to interpret the models. The engine is implemented as a platform-independent web application using the 3D JavaScript library *three.js*[36] and the VR/AR immersive web standard *WebXR* [7]. 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 $\mathbf{SR_{11}}$ (cf. Section 5.1). 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 through triggers, conditions, and actions ($\mathbf{SR_6}$) (cf. Section 5.1).

At the initialization of the *AR Engine*, the application requires an ADOxx XML file containing the model instances representing the *ARWFML* model for importation. Since *ADOxx* provides *XML* files, and web technologies are mostly using *JSON*, the *ADOxx* XML model instance structure was translated into a TypeScript class structure. At initialization, the XML file is translated into this class structure. Subsequently, the various instances are mapped into the different *ARWFML* concepts. The images to detect and the 3D objects are loaded into the 3D environment in a background process.

In order to generate the AR workflow, which is defined by the *ARWFML* model and imported at initialization, the entire workflow is translated into a custom state machine.[37] A state distinguishes the concept it addresses, such as *Origin*, *Condition*, or *Statechange*. The workflow logic is then built by triggering one or multiple states (*Statechanges* or *Conditions*) in accordance with the *FlowScene* model after the currently activated state is set to "done".

Each state in the state machine is associated with a state attribute, which can be set to one of three values: *active*, *inactive*, or *done*. On entering the state, the value is set to "active". The distinction between the type of states and its instance in the *FlowScene* model is made possible by setting the attributes *currentOrigin*, *currentConditions*, and *currentStatechange* after entering the state. Furthermore, the properties *nextConditions* and *nextStatechanges* are set after entering the state. They contain the next *Conditions* to observe, or the next *Statechange* to trigger. If a condition is met or a *Statechange* has been visualized, the state is set to "done" and the next state is triggered.

Given that a *FlowScene* is not a linear workflow, but may have multiple active conditions or triggered *Statechanges* occurring simultaneously, composite state machines are used. By creating such a state machine, the entirety of the *FlowScene* is represented in a single structure that can be efficiently traversed by the animation loop of the AR application. Thereby, the state machine describes the operational semantics of the *ARWFML* language for the implementation of the *AR Engine* [81].

After the initiation of the state machine, the first state is designated as *active* and the user is then able to start the AR session by clicking a button. All *Detectables* defined in the ObjectSpace model are tracked in every frame. By detecting these *Detectables* and considering other concepts defined in *Conditions*, for example, timers or sensor data, the user is guided through the AR workflow. Consequently, the attributes of activated *Statechanges* are applied to the previously loaded 3D objects (*Augmentations*), including visibility, position, and rotation. It should be noted that the WebXR "image-tracking" feature is only available on mobile devices and not on head-mounted displays. This is due to privacy concerns regarding camera streams, which is currently disabled on most HMDs.

Fig. 12 shows an example of the *AR Engine* on the furniture assembly use case introduced in Section 6.2. In the upper left, the *ADOxx ARWFML* FlowScene is depicted, thereby providing a clear representation of the workflow model. The top right image depicts the initial six states in the state machine, generated following the import of the ADOxx XML file, offering insight into the initial workflow structure. Initially, the only state that is *active* is the "Origin" state. The execution of the models of the use case is shown at the bottom of Fig. 12 by using parts from an IKEA table [82]. The screenshots were taken while using the WebXR *AR Engine* in the Chrome browser on a *Samsung Galaxy Tab S7* tablet. The first image shows the origin marker in the real world that must be detected on application start-up. The second image shows the *Statechange* "Init MiddlePlate". It superimposes the 3D object of *MiddlePlate* on top of the real *MiddlePlate*, whose existence, position, and orientation are detected via a marker — *Detectable*. The third image shows the *Statechange* "Leg 1 Positioned". The augmentation shows where the next leg shall be attached. Several colored markers are placed on the real objects at strategic points and according to the *ObjectSpace* model. Once a marker is detected, it is decided based on the current state of the state machine and the *FlowScene* model if it triggers a *Statechange* or not. If a *Statechange* is triggered, the workflow moves on and waits until the next *Condition* in line is met. The flexible structure of the *ARWFML* allows multiple workflow paths to be active at the same time by checking for multiple *Conditions* simultaneously. To avoid making the example unnecessarily complex, the concepts of *Resolves* and *Observer* were not used, and there is only a sequential workflow.

8.2. AR engine implementation based on MM-AR

As concluded after design cycle two (DC2) in Section 6, *ADOxx* is not suitable for modeling 3D models. Therefore, the *ARWFML* has been implemented on the new metamodeling platform — see Section 7. After the development of the *MM-AR* platform and the implementation of the *ARWFML* on *MM-AR*, in design cycle four the first prototype of the *AR Engine* was adapted to be compatible with the *MM-AR* metamodeling language. Since *MM-AR* relies on the *Global Shared Datastructure* module (see Section 7), all model

[36] https://github.com/mrdoob/three.js/

[37] https://www.omg.org/spec/UML/2.5.1/PDF

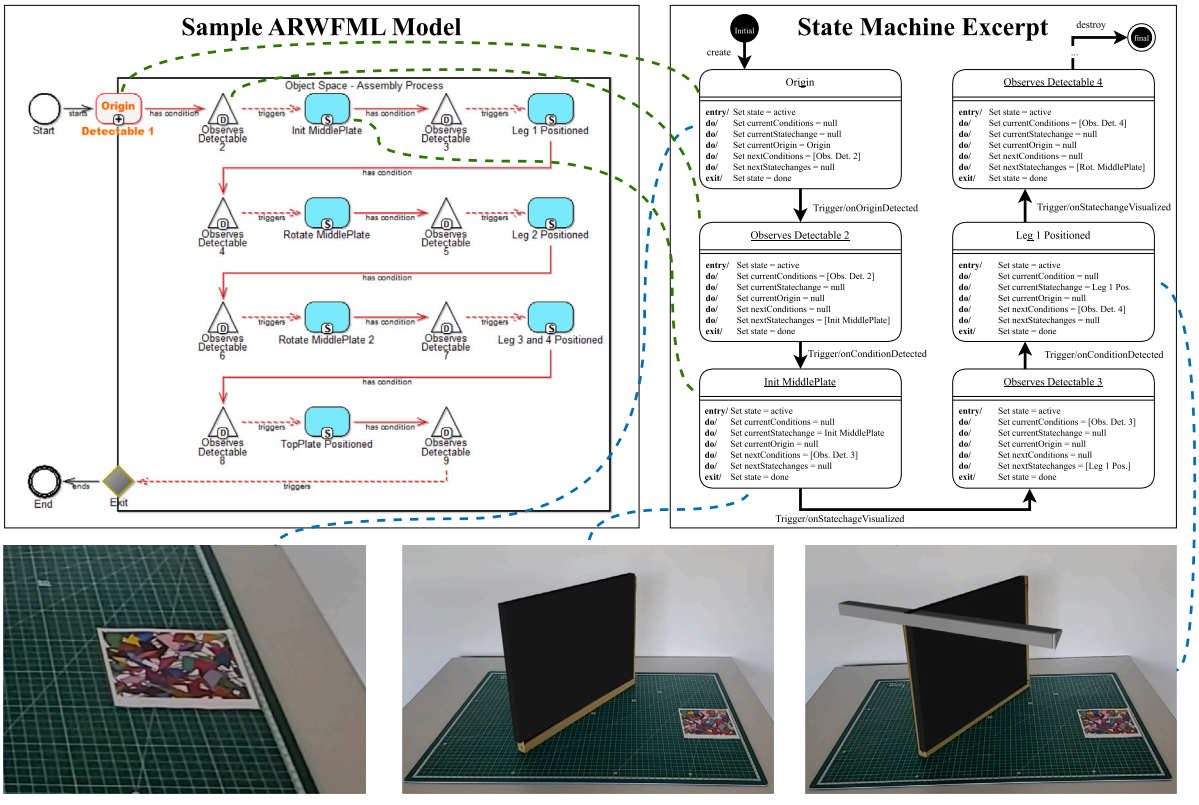


Fig. 12. Example of the *AR Engine* on the furniture assembly use case. (Top left) ARWFML FlowScene in ADOxx. (Top right) Excerpt of the *State Machine* when importing the ADOxx XML file. (Bottom) Screenshot of the running *AR Engine* at the origin and the first two *Statechanges* of the workflow. The dotted lines visualize some exemplary relations between the state machine and the *ARWFML* model and the running AR application.

instances are already typed and do not require mapping during initialization. In comparison to the *AR Engine* v1.0, the components of the state machine for *AR Engine* v2.0 did not change, but the assigned data types are from the *Global Shared Datastructure* of the *MM-AR* platform. Accordingly, the data exported from the *MM-AR* platform can be executed directly on the *AR Engine*.

AR Engine v2.0 was implemented for an evaluation project of *ARWFML* on the new platform, but has not been directly integrated into the *MM-AR* metamodeling platform. It is based on the same technology and structure as *MM-AR*, so it could be integrated as a module in the web client of the platform.

8.3. Evaluation of design cycle four: Comparison of AR engines

There are different paths one could take to develop AR applications with the *ARWFML*. Thus, in the final evaluation, we compare the different paths against each other. Fig. 13 shows the overview of the different paths of the *AR Engine* implementations. Thereby, we consider four different dimensions. (a) The *Metamodeling Language* used as base for the *ARWFML* models. (b) The *Export Format* resulting from the models. (c) The *Technology Stack* used for the development of the *AR Engine*, and (d) the *Execution Platform* supported by the *AR Engine*.

Considering the first dimension, there are only two implementations on modeling tools, *ADOxx* and *MM-AR*, that allow for the modeling of *ARWFML* models. Thus, the first dimension has only this two possibilities. Since ADOxx supports the XML export format, and the *MM-AR* supports the JSON format, these are the two possibilities on the second dimension. Regarding the technology stack for the creation of AR applications, there are different possibilities. Both implementations of the *AR Engine* are based on WebXR. However, there are other development platforms, such as the introduced Unity framework or Unreal Engine — cf. Section 3.2. Since both *AR Engines* were implemented with WebXR, there are only two supported paths. The fourth dimension is about the different platform to execute the final AR application. There are various handheld devices and HMDs for the execution of AR applications. However, the most popular are Android and iOS handhelds, as well as Android HMDs such as the different Meta Quests[38] or the Apple Vision Pro.[39]

[38] https://www.meta.com/quest/

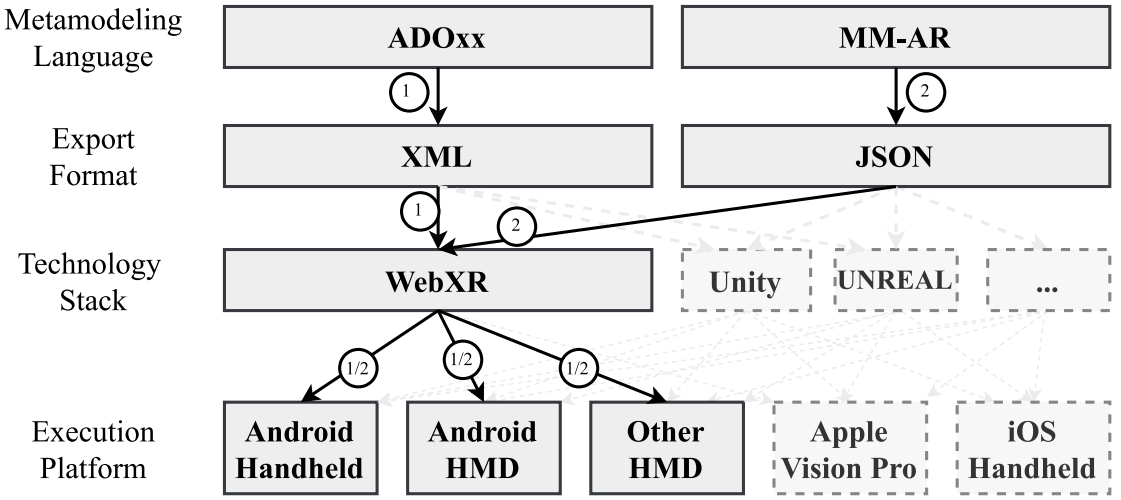


Fig. 13. Overview of the different paths for *AR Engines*. (1) First *AR Engine* v1.0 based on ADOxx and WebXR. (2) *AR Engine* v2.0 based on the *MM-AR* platform and WebXR. Dotted lines signify limitations in support due to external restrictions or not implemented solutions.

Comparing the two versions of the *AR Engine*, there is no difference in the final execution of the AR application. Both of them are using WebXR as ground technology and thus, support all WebXR compatible devices, such as Android-based handhelds and HMDs, as well as partially the HMD Apple *Vision Pro*. In regard to the metamodeling language used for the approach, *AR Engine* v1.0 uses the ADOxx metamodeling language, while v2.0 is based on *MM-AR*. As discussed above, the ADOxx modeling platform is not suitable for 3D modeling, and the export format (XML) is not directly compatible with the WebXR technology. Thus, *AR Engine* v2.0 is better compatible with 3D technologies. In regard to WebXR and its device support, it must be noted, that HMDs in general support WebXR sessions. However, due to privacy reasons, there is currently no possibility to access the camera stream during an AR session. Thus, HMDs currently do not support the detection of images or 3D objects in WebXR. Furthermore, iOS handhelds currently do not support WebXR and the Apple Vision Pro only supports WebXR VR sessions, but not AR sessions. This reduces the usability of the two *AR Engines* to Android-based tablets and smartphones. We are aware that WebXR in isolation does not offer sufficient breadth in terms of implementation. However, given that *MM-AR* is browser-based by default and thus provides seamless WebXR integration across all major browsers, it was deemed appropriate to use the WebXR API for both *AR Engines*, despite the fact that the WebXR is not yet available on all HMDs due to regulatory privacy concerns. In addition, an implementation of a *Unity AR Engine* for the Apple Vision Pro has been made. However, it should be noted that this implementation is not publication-ready. Thus, this additional path is not considered in Fig. 13.

Since the *MM-AR* platform provides the *Global Shared Datastructure* module, and the *MM-AR* is much more convenient for modeling with *ARWFML* models (cf. Section 7.6), the *MM-AR* metamodeling language and *ARWFML* models in JSON format are the preferred path. Instead of using WebXR, it would also be possible to develop an *AR Engine* on the basis of another technology stack, e.g., Unity or the Unreal Engine. In general, Unity and Unreal Engine are very flexible in regard to the execution platform. However, as mentioned above, when having specific requirements, such as the detection of images or 3D objects, specific frameworks are needed. These frameworks are mostly for specific devices, e.g., by using the Apple AR Kit,[40] only iOS handhelds and the Apple Vision Pro would be supported, which is not conform to $\mathbf{SR_{11}}$ and $\mathbf{SR_{12}}$.

8.3.1. Summary of evaluation in the fourth design cycle

In conclusion, *AR Engine* v2.0 is the preferable option in comparison to *AR Engine* v1.0, given that it is built on the same technological foundation as the *MM-AR* platform, thereby allowing for direct integration. At this stage of development, there is no discernible advantage to be gained from utilizing a professional development platform such as Unity or Unreal Engine for the development of the *AR Engine* in comparison to the WebXR implementations. Moreover, given that Unity and Unreal Engine employ different technologies than the *MM-AR* platform, *AR Engine* v2.0 is the optimal choice at this point. Should a generic approach to utilizing 3D object detection emerge on another development platform, this could alter the choice in the future.

9. Conclusion and outlook

The definition of AR applications is a growing topic and is expected to become even more relevant with the global availability of new, lightweight HMDs as the technology advances.

[39] https://www.apple.com/apple-vision-pro/
[40] https://developer.apple.com/documentation/arkit

In this paper, we presented the entire research process for the development of a new approach that allows the modeling and execution of complex augmented reality workflows for diverse application scenarios without programming knowledge. We could derive the requirements for such a modeling language to answer research question one (RQ1). Subsequently, we could design the modeling language using design science principles in four cycles thus addressing research question two (RQ2), and finally evaluate the language for answering research question three (RQ3). The proposed modeling language *ARWFML* permits designers to specify three distinct categories of visual models: (1) those utilized for defining the AR environment, (2) the AR workflow, and (3) the various statechanges occurring within this workflow. Consequently, the language emphasizes a high degree of abstraction and separation of concerns. This abstraction serves to bridge potential gaps in knowledge regarding the technical implementation of AR environments, thereby enabling users to concentrate on the content and functionality of AR applications. The technical feasibility was demonstrated by implementing the modeling language using the ADOxx platform. Due to limitations of traditional 2D modeling platforms, a new web-based, multi-dimensional, modeling environment denoted as *MM-AR* has been developed and the *ARWFML* has been adapted accordingly. The new platform is based on the 3D JavaScript library *THREE.js* and the mixed reality immersive web standard *WebXR* [7]. The defined *ARWFML* models can then be executed with an *AR Engine* on any WebXR supported device. Moreover, alternative development platforms have been considered, yet they currently do not offer advantages that outweigh those of the WebXR approach.

In addition to conceptual and technical contributions, this paper demonstrates how modeling languages for the domain of augmented reality can be evaluated from multiple perspectives. These include a comparative evaluation against requirements, technical implementations, and workflow capabilities; a comparison against other modeling tools; and an empirical comprehensibility study of the 3D-based models of *ARWFML*.

In summary, the *ARWFML* is more expressive than comparable approaches. 3D modeling approaches require adequate tooling. This has been addressed by the development of *MM-AR*, which is the first 3D-enabled conceptual modeling tool. In addition, a empirical study showed, that the *ARWFML* constructs are comprehensible and a demonstration showed that the defined models are actually executable via an *AR Engine*.

This research project also has some limitations. First, the use cases presented in Section 6.2 are relatively simple. Second, the implementation has not yet undergone an empirical evaluation through a usability study. Third, the *AR Engine* is limited to WebXR, which has currently privacy limitations on HMDs. Lastly, the *AR Engine* is currently not integrated in the modeling environment, but is a separate web application.

Thus, in future iterations, we plan to advance the usability of the *MM-AR* modeling platform so that further empirical studies can be conducted with it. Furthermore, the integration of the *AR Engine* into *MM-AR* is envisaged, and the *ARWFML* should be extended towards supporting further workflow patterns.

CRediT authorship contribution statement

Fabian Muff: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Hans-Georg Fill:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the *Smart Living Lab* (https://www.smartlivinglab.ch/en/), funded by the University of Fribourg, EPFL, and HEIA-FR. The sponsors had no role in the research design, data collection, analysis, interpretation, manuscript writing, or decision to submit for publication.

Data availability

No data was used for the research described in the article.

References

[1] J.S. Roo, M. Hachet, One reality: Augmenting how the physical world is experienced by combining multiple mixed reality modalities, in: K. Gajos, J. Mankoff, C. Harrison (Eds.), Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, UIST 2017, Quebec City, QC, Canada, October 22 – 25, 2017, ACM, 2017, pp. 787–795, http://dx.doi.org/10.1145/3126594.3126638.

[2] F. Zhou, H.B. Duh, M. Billinghurst, 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, UK, 15-18th September 2008, IEEE Computer Society, 2008, pp. 193–202, http://dx.doi.org/10.1109/ISMAR.2008.4637362.

[3] R.T. Azuma, A survey of augmented reality, Presence Teleoperators Virtual Env. 6 (4) (1997) 355–385, http://dx.doi.org/10.1162/pres.1997.6.4.355.

[4] D. Schmalstieg, T. Höllerer, Augmented Reality Principles and Practice, Addison-Wesley, Boston, 2016.

[5] F. Wild, P. Scott, P. Lefrere, J. Karjalainen, K. Helin, A. Naeve, E. Isaksson, Towards data exchange formats for learning experiences in manufacturing workplaces, in: M. Kravcik, A. Mikroyannidis, V. Pammer, M. Prilla, T.D. Ullmann, F. Wild (Eds.), Proceedings of the 4th Workshop on Awareness and Reflection in Technology-Enhanced Learning, ARTEL@EC-TEL 2014, Graz, Austria, September 16, 2014, in: CEUR Workshop Proceedings, 1238, CEUR-WS.org, 2014, pp. 23–33, URL: http://ceur-ws.org/Vol-1238/paper2.pdf.
[6] K. Yin, Z. He, J. Xiong, J. Zou, K. Li, S.-T. Wu, Virtual reality and augmented reality displays: advances and future perspectives, J. Phys.: Photonics 3 (2) (2021) 022010, http://dx.doi.org/10.1088/2515-7647/abf02e.
[7] B. Jones, M. Goregaokar, R. Cabanier, WebXR Device API, W3C Candidate Recommendation Draft, work in progress, World Wide Web Consortium, 2026, URL: https://www.w3.org/TR/2026/CRD-webxr-20260609/. in preparation.
[8] T. Nguyen, E. Brown, K. Chauhan, B. Lee, J. Erensen, Emerging tech: Revenue opportunity projection for AR and VR head-mounted displays, 2024, URL: https://www.gartner.com/en/documents/5179263. (Date accessed 18 November 2024).
[9] R. Hernandez, R. Emanuelle, Immersive technology trends – the tipping point is near, 2024, URL: https://www.pwc.com/us/en/tech-effect/emerging-tech/immersive-technology-trends-in-2024.html. (Date accessed 18 Novenmber 2024).
[10] G. Grambow, D. Hieber, R. Oberhauser, C. Pogolski, A context and augmented reality BPMN and BPMS extension for industrial internet of things processes, in: A. Marrella, B. Weber (Eds.), BPM Workshops, in: Lecture Notes in Business Information Processing, vol. 436, Springer, Cham, 2021, pp. 379–390, http://dx.doi.org/10.1007/978-3-030-94343-1_29.
[11] D. Rumiński, K. Walczak, CARL: A language for modelling contextual augmented reality environments, in: L.M. Camarinha-Matos, N.S. Barrento, R. Mendon¸ca (Eds.), Technological Innovation for Collective Awareness Systems, in: IFIP Advances in Information and Communication Technology, vol. 423, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 183–190, http://dx.doi.org/10.1007/978-3-642-54734-8_21.
[12] M. Lechner, ARML 2.0 in the context of existing AR data formats, in: M.E. Latoschik, D. Reiners, R. Blach, P.A. Figueroa, C.A. Wingrave (Eds.), 6th Workshop on Software Engineering and Architectures for Realtime Interactive Systems, SEARIS 2013, Orlando, FL, USA, March 17, 2013, IEEE Computer Society, 2013, pp. 41–47, http://dx.doi.org/10.1109/SEARIS.2013.6798107.
[13] I. Ruiz-Rube, R.B. Pérez, J.M. Mota, I. Arnedillo-Sánchez, Model-driven development of augmented reality-based editors for domain specific languages, IxD&A (2020) 18, http://dx.doi.org/10.55612/s-5002-045-011.
[14] R. Campos-López, E. Guerra, J. de Lara, Towards automating the construction of augmented reality interfaces for information systems, in: E. Insfrán, S.A. ao, M. Fernández, C. Barry, M. Lang, H. Linger, C. Schneider (Eds.), Information Systems Development: Crossing Boundaries Between Development and Operations (DevOps) in Information Systems (ISD2021 Proceedings), Valencia, Spain, September 8–10, 2021, Universitat Politècnica de València, 2021, URL: https://aisel.aisnet.org/isd2014/proceedings2021/hci/6.
[15] S. Gregor, A.R. Hevner, Positioning and presenting design science research for maximum impact, MIS Q. 37 (2) (2013) 337–355, URL: https://aisel.aisnet.org/misq/vol37/iss2/3.
[16] F. Muff, H. Fill, Past achievements and future opportunities in combining conceptual modeling with VR/AR: A systematic derivation, in: B. Shishkov (Ed.), Business Modeling and Software Design - 13th International Symposium, BMSD 2023, Utrecht, the Netherlands, July 3–5, 2023, Proceedings, in: Lecture Notes in Business Information Processing, vol. 483, Springer, 2023, pp. 129–144, http://dx.doi.org/10.1007/978-3-031-36757-1_8.
[17] D. Karagiannis, H.C. Mayr, J. Mylopoulos (Eds.), Domain-Specific Conceptual Modeling, Concepts, Methods and Tools, Springer, 2016, http://dx.doi.org/10.1007/978-3-319-39417-6.
[18] D. Karagiannis, M. Lee, K. Hinkelmann, W. Utz (Eds.), Domain-Specific Conceptual Modeling: Concepts, Methods and ADOxx Tools, Springer International Publishing, Cham, 2022, http://dx.doi.org/10.1007/978-3-030-93547-4, URL: https://link.springer.com/10.1007/978-3-030-93547-4.
[19] X. Boucher, R.A. Buchmann, H.-G. Fill, D. Kyritsis, W. Utz (Eds.), Domain-Specific Conceptual Modeling: The OMiLAB Community of Practice, Springer Nature Switzerland, Cham, 2026, http://dx.doi.org/10.1007/978-3-031-98660-4, URL: https://link.springer.com/10.1007/978-3-031-98660-4.
[20] F. Härer, H. Fill, Past trends and future prospects in conceptual modeling - A bibliometric analysis, in: ER'2020, Springer, 2020, pp. 34–47, http://dx.doi.org/10.1007/978-3-030-62522-1_3.
[21] N. Thuan, A. Drechsler, P. Antunes, Construction of design science research questions, Commun. Assoc. Inf. Syst. 44 (1) (2019) http://dx.doi.org/10.17705/1CAIS.04420, URL: https://aisel.aisnet.org/cais/vol44/iss1/20.
[22] F. Muff, H.-G. Fill, Multi-faceted evaluation of modeling languages for augmented reality applications the case of arwfml, in: W. Maass, H. Han, H. Yasar, N. Multari (Eds.), Conceptual Modeling, Springer Nature Switzerland, Cham, 2025, pp. 75–93, http://dx.doi.org/10.1007/978-3-031-75872-0_5.
[23] F. Muff, H. Fill, A domain-specific visual modeling language for augmented reality applications using WebXR, in: J. ao Paulo A. Almeida, J. Borbinha, G. Guizzardi, S. Link, J. Zdravkovic (Eds.), Conceptual Modeling - 42nd International Conference, ER 2023, Lisbon, Portugal, November 6–9, 2023, Proceedings, in: Lecture Notes in Computer Science, vol. 14320, Springer, 2023, pp. 334–353, http://dx.doi.org/10.1007/978-3-031-47262-6_18.
[24] F. Muff, H. Fill, 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), ACM, 2024, http://dx.doi.org/10.1145/3652620.3687779.
[25] V.K. Vaishnavi, B. Kuechler, Design science research in information systems, 2004,2021, URL: http://www.desrist.org/design-research-in-information-systems/. Last updated December 15, 2023.
[26] V.K. Vaishnavi, W. Kuechler, Design Science Research Methods and Patterns: Innovating Information and Communication Technology, 2nd Edition, second ed., CRC Press, 2015, http://dx.doi.org/10.1201/b18448.
[27] A.R. Hevner, S.T. March, J. Park, S. Ram, Design science in information systems research, MIS Q. 28 (1) (2004) 75–105, URL: http://misq.org/design-science-in-information-systems-research.html.
[28] K. Peffers, T. Tuunanen, M.A. Rothenberger, S. Chatterjee, A design science research methodology for information systems research, J. Manage. Inf. Syst. 24 (3) (2008) 45–77, http://dx.doi.org/10.2753/MIS0742-1222240302.
[29] J. Venable, J. Pries-Heje, R. Baskerville, FEDS: a framework for evaluation in design science research, Eur. J. Inf. Syst. 25 (1) (2016) 77–89, http://dx.doi.org/10.1057/ejis.2014.36.
[30] G. Karsai, H. Krahn, C. Pinkernell, B. Rumpe, M. Schneider, S. Völkel, Design guidelines for domain specific languages, in: M. Rossi, J. Sprinkle, J. Gray, J.-P. Tolvanen (Eds.), in: Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM '09), Orlanda, vol. B-108, Helsingin Kauppakorkeakoulu, 2009, pp. 7–13, URL: http://www.dsmforum.org/events/DSM09/Papers/Karsai.pdf.
[31] U. Frank, Domain-specific modeling languages: Requirements analysis and design guidelines, in: I. Reinhartz-Berger, A. Sturm, T. Clark, S. Cohen, J. Bettin (Eds.), Domain Engineering, Product Lines, Languages, and Conceptual Models, Springer, 2013, pp. 133–157, http://dx.doi.org/10.1007/978-3-642-36654-3_6.
[32] S. Jannaber, D.M. Riehle, P. Delfmann, O. Thomas, J. Becker, Designing a framework for the development of domain-specific process modelling languages, in: A. Maedche, J. vom Brocke, A.R. Hevner (Eds.), Designing the Digital Transformation - 12th International Conference, DESRIST 2017, Karlsruhe, Germany, May 30 - June 1, 2017, Proceedings, in: Lecture Notes in Computer Science, vol. 10243, Springer, 2017, pp. 39–54, http://dx.doi.org/10.1007/978-3-319-59144-5_3.
[33] N. Visic, H. Fill, R.A. Buchmann, D. Karagiannis, 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, Greece, May 13–15, 2015, IEEE, 2015, pp. 286–297, http://dx.doi.org/10.1109/RCIS.2015.7128889.
[34] K. Siau, M. Rossi, Evaluation techniques for systems analysis and design modelling methods - a review and comparative analysis, Inf. Syst. J. 21 (3) (2011) 249–268, http://dx.doi.org/10.1111/j.1365-2575.2007.00255.x.

[35] H. Fill, D. Karagiannis, On the conceptualisation of modelling methods using the ADOxx meta modelling platform, Enterp. Model. Inf. Syst. Arch. Int. J. Concept. Model. 8 (1) (2013) 4–25, http://dx.doi.org/10.18417/emisa.8.1.1.
[36] W.M.P. van der Aalst, A.H.M. ter Hofstede, Workflow patterns: On the expressive power of (Petri-net-based) workflow languages, DAIMI / PB 560 (2002) 1–20, http://dx.doi.org/10.7146/dpb.v31i560.7117.
[37] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, A.P. Barros, Workflow patterns, Distrib. Parallel Databases 14 (1) (2003) 5–51, http://dx.doi.org/10.1023/A:1022883727209.
[38] D.L. Moody, The method evaluation model: a theoretical model for validating information systems design methods, in: C.U. Ciborra, R. Mercurio, M. de Marco, M. Martinez, A. Carignani (Eds.), Proceedings of the 11th European Conference on Information Systems, ECIS 2003, Naples, Italy 16–21 June 2003, 2003, pp. 1327–1336, URL: http://aisel.aisnet.org/ecis2003/79.
[39] R. Doerner, W. Broll, P. Grimm, B. Jung (Eds.), Virtual and Augmented Reality (VR/AR): Foundations and Methods of Extended Realities (XR), Springer International Publishing, 2022, http://dx.doi.org/10.1007/978-3-030-79062-2.
[40] D. Saxena, J.K. Verma, Recreating reality: Classification of computer-assisted environments, in: J.K. Verma, S. Paul (Eds.), Advances in Augmented Reality and Virtual Reality, Springer Singapore, Singapore, 2022, pp. 3–9, http://dx.doi.org/10.1007/978-981-16-7220-0_1.
[41] J. Gray, B. Rumpe, The evolution of model editors: browser- and cloud-based solutions, Softw. Syst. Model. 15 (2) (2016) 303–305, http://dx.doi.org/10.1007/S10270-016-0524-2.
[42] M. Maróti, T. Kecskés, R. Kereskényi, B. Broll, P. Völgyesi, L. Jurácz, T. Levendovszky, Á. Lédeczi, Next generation (meta)modeling: Web- and cloud-based collaborative tool infrastructure, in: D. Balasubramanian, C. Jacquet, P.V. Gorp, S. Kokaly, T. Mészáros (Eds.), Proceedings of the 8th Workshop on Multi-Paradigm Modeling Co-Located with the 17th International Conference on Model Driven Engineering Languages and Systems, MPM@MODELS 2014, Valencia, Spain, September 30, 2014, in: CEUR Workshop Proceedings, 1237, CEUR-WS.org, 2014, pp. 41–60, URL: https://ceur-ws.org/Vol-1237/paper5.pdf.
[43] E. Syriani, H. Vangheluwe, R. Mannadiar, C. Hansen, S.V. Mierlo, H. Ergin, AToMPM: A web-based modeling environment, in: Y. Liu, S. Zschaler, B. Baudry, S. Ghosh, D.D. Ruscio, E.K. Jackson, M. Wimmer (Eds.), 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, USA, September 29 - October 4, 2013, in: CEUR Workshop Proceedings, vol. 1115, CEUR-WS.org, 2013, pp. 21–25, URL: https://ceur-ws.org/Vol-1115/demo4.pdf.
[44] H. Metin, D. Bork, Introducing BIGUML: A flexible open-source GLSP-based web modeling tool for UML, in: ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2023 Companion, VästeråS, Sweden, October 1–6, 2023, IEEE, 2023, pp. 40–44, http://dx.doi.org/10.1109/MODELS-C59198.2023.00016.
[45] The Value Engineers, e3value online tool 1.8.56, 2024, https://e3web.thevalueengineers.nl. (Accessed 05 February 2024).
[46] BOC Group, ADONIS community edition, 2024, https://www.adonis-community.com/en/. (Accessed 05 February 2024).
[47] SAP, Signavio process manager, 2024, https://www.signavio.com/products/process-manager/. (Accessed 05 February 2024).
[48] BOC Group, ADOit community edition, 2024, https://www.adoit-community.com/en/. (Accessed 05 February 2024).
[49] A.C. Bock, U. Frank, Low-code platform, Bus. Inf. Syst. Eng. 63 (6) (2021) 733–740, http://dx.doi.org/10.1007/s12599-021-00726-8.
[50] D. Di Ruscio, D.S. Kolovos, J. de Lara, A. Pierantonio, M. Tisi, M. Wimmer, Low-code development and model-driven engineering: Two sides of the same coin? Softw. Syst. Model. 21 (2) (2022) 437–446, http://dx.doi.org/10.1007/s10270-021-00970-2.
[51] S. Curty, F. Härer, H. Fill, Design of blockchain-based applications using model-driven engineering and low-code/no-code platforms: a structured literature review, Softw. Syst. Model. 22 (6) (2023) 1857–1895, http://dx.doi.org/10.1007/S10270-023-01109-1.
[52] P. Nicolaescu, M. Rosenstengel, M. Derntl, R. Klamma, M. Jarke, Near real-time collaborative modeling for view-based web information systems engineering, Inf. Syst. 74 (Part) (2018) 23–39, http://dx.doi.org/10.1016/J.IS.2017.07.008.
[53] J. Ladleif, A. von Weltzien, M. Weske, chor-js: A modeling framework for BPMN 2.0 choreography diagrams, in: J.I. Panach, R.S.S. Guizzardi, D.B. Claro (Eds.), Proceedings of the ER Forum and Poster & Demos Session 2019, in: CEUR Workshop Proceedings, vol. 2469, CEUR-WS.org, 2019, pp. 113–117, URL: https://ceur-ws.org/Vol-2469/ERDemo02.pdf.
[54] D. Ruminski, K. Walczak, Dynamic composition of interactive AR scenes with the CARL language, in: IISA 2014, the 5th International Conference on Information, Intelligence, Systems and Applications, IEEE, Chania, Crete, Greece, 2014, pp. 329–334, http://dx.doi.org/10.1109/IISA.2014.6878808.
[55] F. Wild, C. Perey, B. Hensen, R. Klamma, IEEE standard for augmented reality learning experience models, in: IEEE International Conference on Teaching, Assessment, and Learning for Engineering, TALE 2020, IEEE, 2020, pp. 1–3, http://dx.doi.org/10.1109/TALE48869.2020.9368405.
[56] R. Seiger, R. Kühn, M. Korzetz, U. Aßmann, HoloFlows: modelling of processes for the Internet of Things in mixed reality, Softw. Syst. Model. 20 (5) (2021) 1465–1489, http://dx.doi.org/10.1007/s10270-020-00859-6.
[57] R. Campos-López, E. Guerra, J. de Lara, A. Colantoni, A. Garmendia, Model-driven engineering for augmented reality, J. Object Technol. 22 (2) (2023) 1–15, http://dx.doi.org/10.5381/JOT.2023.22.2.A7.
[58] L. Brunschwig, R. Campos-Lopez, E. Guerra, J. de Lara, 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), IEEE, Madrid, ES, 2021, pp. 56–60, http://dx.doi.org/10.1109/ICSE-NIER52604.2021.00020.
[59] M. Lenk, A. Vitzthum, B. Jung, Model-driven iterative development of 3D web-applications using SSIML, X3D and JavaScript, in: Proceedings of the 17th International Conference on 3D Web Technology - Web3D '12, ACM Press, Los Angeles, California, 2012, p. 161, http://dx.doi.org/10.1145/2338714.2338742, URL: http://dl.acm.org/citation.cfm?doid=2338714.2338742.
[60] M. Mernik, J. Heering, A.M. Sloane, When and how to develop domain-specific languages, ACM Comput. Surv. 37 (4) (2005) 316–344, http://dx.doi.org/10.1145/1118890.1118892.
[61] J. Gulden, E.S.K. Yu, Toward Requirements-Driven Design of Visual Modeling Languages, in: R.A. Buchmann, D. Karagiannis, M. Kirikova (Eds.), The Practice of Enterprise Modeling - 11th IFIP WG 8.1. Working Conference, PoEM 2018, Vienna, Austria, October 31 – November 2, 2018, Proceedings, in: Lecture Notes in Business Information Processing, vol. 335, Springer, 2018, pp. 21–36, http://dx.doi.org/10.1007/978-3-030-02302-7_2.
[62] H.-G. Fill, F. Härer, F. Muff, S. Curty, Towards augmented enterprise models as low-code interfaces to digital systems, in: B. Shishkov (Ed.), Business Modeling and Software Design, in: Lecture Notes in Business Information Processing, vol. 422, Springer International Publishing, Cham, 2021, pp. 343–352, http://dx.doi.org/10.1007/978-3-030-79976-2_22.
[63] D. Götzinger, E. Miron, F. Staffel, OMiLAB: An open collaborative environment for modeling method engineering, in: D. Karagiannis, H.C. Mayr, J. Mylopoulos (Eds.), Domain-Specific Conceptual Modeling, Concepts, Methods and Tools, Springer, 2016, pp. 55–76, http://dx.doi.org/10.1007/978-3-319-39417-6_3.
[64] H. Fill, T. Redmond, D. Karagiannis, Formalizing meta models with FDMM: The ADOxx case, in: J. Cordeiro, L.A. Maciaszek, J. Filipe (Eds.), Enterprise Information Systems - 14th International Conference, ICEIS 2012, Wroclaw, Poland, June 28 – July 1, 2012, Revised Selected Papers, in: Lecture Notes in Business Information Processing, vol. 141, Springer, 2012, pp. 429–451, http://dx.doi.org/10.1007/978-3-642-40654-6_26.
[65] D.L. Moody, The physics of notations: Toward a scientific basis for constructing visual notations in software engineering, IEEE Trans. Softw. Eng. 35 (6) (2009) 756–779, http://dx.doi.org/10.1109/TSE.2009.67.
[66] D. Bork, B. Roelens, A technique for evaluating and improving the semantic transparency of modeling language notations, Softw. Syst. Model. 20 (4) (2021) 939–963, http://dx.doi.org/10.1007/s10270-021-00895-w.

[67] H. Fill, T. Redmond, D. Karagiannis, FDMM: A formalism for describing ADOxx meta models and models, in: L.A. Maciaszek, A. Cuzzocrea, J. Cordeiro (Eds.), ICEIS 2012 - Proceedings of the 14th International Conference on Enterprise Information Systems, Volume 3, Wroclaw, Poland, 28 June – 1 July, 2012, SciTePress, 2012, pp. 133–144, http://dx.doi.org/10.5220/0003971201330144.
[68] F. Muff, H. Fill, ADOxx library and UseCase models for the ER23 publication: A domain-specific visual modeling language for augmented reality applications using WebXR, 2023, http://dx.doi.org/10.5281/zenodo.8207639.
[69] C. Petri, W. Reisig, Petri net, Scholarpedia 3 (4) (2008) 6477, http://dx.doi.org/10.4249/scholarpedia.6477.
[70] F. Muff, H. Fill, 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, Canada, October 18-21, 2021, in: CEUR Workshop Proceedings, vol. 2958, CEUR-WS.org, 2021, pp. 49–54, URL: https://ceur-ws.org/Vol-2958/paper9.pdf.
[71] F. Muff, D. Borcard, H.-G. Fill, MM-AR: a web-based open-source metamodeling platform for spatial conceptual modeling, Softw. Syst. Model. (2026) http://dx.doi.org/10.1007/s10270-025-01351-9.
[72] D. Karagiannis, H. Kühn, Metamodelling platforms, in: K. Bauknecht, A.M. Tjoa, G. Quirchmayr (Eds.), E-Commerce and Web Technologies, Third International Conference, EC-Web 2002, Aix-En-Provence, France, September 2–6, 2002, Proceedings, in: Lecture Notes in Computer Science, vol. 2455, Springer, 2002, p. 182, http://dx.doi.org/10.1007/3-540-45705-4_19.
[73] F. Muff, D. Borcard, H.-G. Fill, MMAR metamodeling platform, 2025, URL: https://github.com/MM-AR/mmar.
[74] BOC GmbH, The ADOxx metamodeling platform, 2022, URL: http://www.adoxx.org/live/. (Date accessed 22 November 2022).
[75] D. Bork, C. Schrüffer, D. Karagiannis, Intuitive understanding of domain-specific modeling languages: Proposition and application of an evaluation technique, in: A.H.F. Laender, B. Pernici, E. Lim, J.P.M. de Oliveira (Eds.), in: Conceptual Modeling - 38th International Conference, ER 2019, Salvador, Brazil, November 4-7, 2019, Proceedings, vol. 11788, Springer, 2019, pp. 311–319, http://dx.doi.org/10.1007/978-3-030-33223-5_26.
[76] G. Charness, U. Gneezy, M.A. Kuhn, Experimental methods: Between-subject and within-subject design, J. Econ. Behav. Organ. 81 (1) (2012) 1–8, http://dx.doi.org/10.1016/j.jebo.2011.08.009.
[77] R. Likert, A technique for the measurement of attitudes, A Technique for the Measurement of Attitudes, Archives of Psychology, (140) New York, 1932, [s.n.].
[78] J. Kekes, The pragmatic idealism of Nicholas Rescher, Philos. Phenomenol. Res. 54 (2) (1994) 391–394, URL: http://www.jstor.org/stable/2108498.
[79] D. Freedman, R. Pisani, R. Purves, Statistics (International Student Edition), fourth ed., WW Norton & Company, New York, 2007,
[80] D.E. Hinkle, W. Wiersma, S.G. Jurs, Applied Statistics for the Behavioral Sciences, (663) Houghton Mifflin, 2003.
[81] D. Harel, B. Rumpe, Meaningful modeling: what's the semantics of "semantics"? Computer 37 (10) (2004) 64–72, http://dx.doi.org/10.1109/MC.2004.172.
[82] IKEA, IKEA online shop, 2023, URL: https://www.ikea.com/us/en/p/knarrevik-nightstand-black-30381183/. (Date accessed 28 April 2023).

Fabian Muff is a former senior researcher and lecturer at the University of Fribourg, Switzerland and works as innovation architect at the swiss government. He holds a Ph.D. from the University of Fribourg in business informatics. His research activities focus on the combination of conceptual modeling and extended reality technologies, as well as the development of the 3D-enabled metamodeling platform MM-AR.

Hans-Georg Fill is a full professor for business informatics at the University of Fribourg and head of the Digitalization and Information Systems Group. Prior to his engagement at the University of Fribourg he held positions in Germany and Austria. He holds a Ph.D. and a habilitation in business informatics from the University of Vienna, Austria. His research interests include metamodeling, augmented and virtual reality as well as generative AI for conceptual modeling.