



# A Domain-Specific Visual Modeling Language for Augmented Reality Applications Using WebXR

Fabian Muff<sup>(✉)</sup> and Hans-Georg Fill

Research Group Digitalization and Information Systems, University of Fribourg,  
Fribourg, Switzerland

{fabian.muff,hans-georg.fill}@unifr.ch

**Abstract.** Augmented reality (AR) is a technology that overlays digital information onto real-world objects using devices like smartphones, tablets, or head-mounted displays to enrich human comprehension and interaction with the physical environment. The creation of AR software applications requires today advanced coding skills, particularly when aiming to realize complex, multifaceted scenarios. As an alternative, we propose a domain-specific visual modeling language for designing AR scenarios, enabling users to define augmentations and AR workflows graphically. The language has been implemented on the ADOxx metamodeling platform, together with a software engine for running the AR applications using the W3C WebXR Device API for web-based augmented reality. The language and the AR application are demonstrated through a furniture assembly use case. In an initial evaluation, we show, via a comprehensive feature comparison, that the proposed language exhibits a more extensive coverage of AR concepts compared to preceding model-based approaches.

**Keywords:** Augmented Reality · Domain-specific modeling language · Metamodeling

## 1 Introduction

Augmented reality (AR) plays an important role in the ongoing convergence of the physical and the digital world [28]. At its core, augmented reality enhances the user's perception by superimposing visual information such as images, videos, or three-dimensional (3D) visualizations onto real-world environments in real time [2, 39]. 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 [32]. AR reverts to markers or detectors of real-world objects to determine their location and orientation in three-dimensional space to accurately map visual information onto

---

Financial support is gratefully acknowledged by the [Smart Living Lab](#) funded by the University of Fribourg, EPFL, and HEIA-FR.

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.

Recent technological advances have made augmented reality affordable via its availability on standard smartphones and tablets [38]. 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 [18]. In terms of industrial applications, market research by Gartner [27] and PwC [7] indicates that AR is a highly promising technology allowing for broad usage in industrial scenarios such as maintenance tasks or training [15].

Creating augmented reality applications requires today advanced programming skills, e.g., for platforms and APIs such as Vuforia<sup>1</sup>, ARKit<sup>2</sup>, Google ARCore<sup>3</sup>, or MRTK<sup>4</sup>. 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 [21, 30] or with a focus on learning experiences [37]; domain-specific languages for creating AR model editors using Vuforia, ARKit, or MRTK [6, 29, 33]; or a BPMN extension for representing process information in AR using the Unity platform [15]. 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<sup>5</sup>, or Adobe Aero<sup>6</sup>. However, these tools are mostly designed for creating a single AR scene or very simple workflows.

What is missing so far is a visual modeling approach that can represent complex AR workflows for diverse application scenarios, that can be easily adapted to new requirements, and that is based on open standards. To facilitate the creation of AR applications that take advantage of the accessibility, portability, interoperability, and openness of the web, we propose a domain-specific modeling language (DSML) based on models conforming to the W3C WebXR Device API recommendation, thereby enabling the definition of different scenarios such as assembly processes, maintenance tasks, or learning experiences. The development of the language follows guidelines for DSML development proposed by Frank [13]. The DSML has been implemented on the ADOxx metamodeling platform and applied to a furniture assembly use case [11]. For a first evaluation, we conduct a feature comparison with similar languages in the area of augmented reality [34].

The remainder of the paper is organized as follows. Section 2 describes fundamental concepts in AR and the most important development platforms for achieving a common understanding. In Sect. 3, we analyze previous related work

<sup>1</sup> <https://library.vuforia.com/>.

<sup>2</sup> <https://developer.apple.com/augmented-reality/arkit/>.

<sup>3</sup> <https://developers.google.com/ar>.

<sup>4</sup> <https://github.com/Microsoft/MixedRealityToolkit-Unity>.

<sup>5</sup> <https://www.unitear.com/>.

<sup>6</sup> <https://adobe.com/products/aero.html>.

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 and implementation in Sect. 4. This is followed by a use case in Sect. 5. In Sect. 6, we evaluate the language through a feature comparison. Finally, in Sect. 7, we conclude the paper and point to future work.

## 2 Foundations

As augmented reality relies on a range of specific techniques from computer vision to achieve the intended user experience, we will briefly explain the most important concepts in the following for ensuring a common understanding.

### 2.1 Augmented Reality

Augmented reality is a technology that allows computer-generated virtual images to be embedded in the real environment [39], thereby creating a three-dimensional alignment between virtual and real objects that allows for interaction in real-time [2].

Augmented reality relies on three core concepts from the field of computer vision [32]: (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 two-dimensional (2D) or 3D sensor information, e.g., from a camera stream or a LiDAR scanner [9, 31]. 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 [32]. 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 [37]. 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.

## 2.2 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](#), [Wikitude](#), [Vuforia](#), [Kudan](#), [Unreal Engine](#), or [Adobe Aero](#). In addition, open source platforms and SDKs are available, such as [Google ARCore](#), [ARToolKit+](#), [OpenXR](#), or [Holokit](#).

An alternative to the above platforms and SDKs is the WebXR Device API [18]. 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<sup>7</sup>, including handheld smartphones and tablets, as well as *head-mounted displays*, e.g., the Microsoft HoloLens 2<sup>8</sup>. Further, WebXR is already included in the WebKit engine used by iOS Safari<sup>9</sup> and will be supported by the Apple Vision Pro<sup>10</sup>. WebXR does not facilitate the development of technical applications, but applications developed with it are more accessible.

## 3 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 [26]. 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 [30] 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.

Wild et al. [37] 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* has been developed [36], which includes a reference

---

<sup>7</sup> <https://caniuse.com/webxr>.

<sup>8</sup> <https://microsoft.com/en-us/hololens>.

<sup>9</sup> <https://github.com/WebKit/WebKit/tree/main/Source/WebCore/Modules/webxr>.

<sup>10</sup> <https://www.apple.com/apple-vision-pro/>.

implementation<sup>11</sup>. It 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.

A similar approach has been developed by Lechner [21]. He proposes the XML-based *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*<sup>12</sup> in the form of an XML grammar.

Ruiz-Rube et al. [29] proposed a model-driven development approach for creating 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*<sup>13</sup>. It only allows for the definition of AR-based modeling applications and not for the definition of other types of AR applications.

Seiger et al. [33] 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. [15] introduced an approach called *BPMN-CARX*. It stands for a solution integrating context-awareness, visual AR support, and process modeling in BPMN of *Industrial Internet of Things* (IIoT) processes. The approach allows 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.

Campos-Lopez et al. [6] and Brunschwig et al. [5] proposed an automated approach for constructing AR-based interfaces for information systems using model-driven and software language engineering principles without the need for coding knowledge. They introduced a model-driven approach for AR interface construction, where the interface is automatically generated from a high-level domain metamodel of the system and includes AR features like augmentations, a mechanism for anchors based on real-world position, or the recognition of barcodes and quick response (QR) codes. Additionally, it is possible to define API calls to be performed upon certain user interactions, e.g., the creation of objects. The approach is mainly designed for modeling systems that use AR, 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<sup>14</sup>.

In summary, approaches exist for (1) generating specific AR applications based on models and schemata, (2) generating AR-based modeling tools based

<sup>11</sup> <https://github.com/WEKIT-ECS/MIRAGE-XR>.

<sup>12</sup> <http://docs.opengeospatial.org/is/12-132r4/12-132r4.html>.

<sup>13</sup> <https://github.com/spi-fm/ARE4DSL>.

<sup>14</sup> <https://alter-ar.github.io/>.

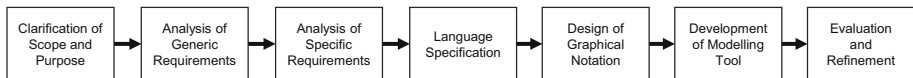
on MDE, and (3) AR modeling applications based on conceptual modeling languages. However, to the best of our knowledge, there is no visual modeling approach available so far for representing executable AR workflows for diverse application scenarios and that is based on open AR standards. Therefore, we 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.

## 4 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 [22]. 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. [13, 19]. 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 [3, 8]. We will thus derive a domain-specific visual modeling language for creating augmented reality applications.

### 4.1 Methodology

Several guidelines and methodologies have been proposed for the development of domain-specific languages, cf. [13, 17, 20, 35]. We will mainly follow the macro process proposed by Frank [13], who describes seven phases including details for each phase - see Fig. 1. For the language specification and the creation of the modeling tool we further considered the methodology by Visic et al. [35], which focuses on the interplay between a modeling language and algorithms and the deployment of the modeling tool.



**Fig. 1.** Seven Phases for Domain-Specific Language Development [13] [p. 8]

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.

### 4.2 Requirements

Frank distinguishes between *generic* and *specific* requirements that need to be analyzed prior to the language specification [13]. As Gulden and Yu pointed

out, these requirements have to be carefully balanced for considering trade-offs between different design alternatives [16], especially in terms of simplicity, comprehensibility, and convenience of use of the language [13].

Thus, we defined the following seven **generic requirements** ( $\mathbf{GR}_{1-7}$ ) for our language as proposed by Frank [13] and in similar fashion by Karsai et al. [20], as well as Jannaber et al. [17]:  $\mathbf{GR}_1$ : 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 Sect. 2, (b) previously identified academic approaches in the area of model-driven engineering for AR [26], and (c) requirements concerning the implementation of the language in terms of satisfying the purpose of platform-independent execution using WebXR [18]. 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 [6, 15, 21, 29, 30, 33, 37]. 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 [6, 29, 37]. 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*) [6, 21, 37].  $\mathbf{SR}_4$ : It must be possible to specify real-world objects that can be tracked during application runtime (*Detectable/Trackable*) [6, 15, 21, 29, 30, 37]. 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. **SR<sub>5</sub>**: Specifying the modification of different objects based on different *actions* is a critical functionality of AR applications [21, 29, 30, 33, 37]. Thus, the modeling language should permit to define transitions to subsequent actions and to directly manipulate and transform augmentations. **SR<sub>6</sub>**: For realizing complex AR workflows [15, 33, 37], *triggers and conditions* are required to enable dynamic branchings in AR applications [6, 15, 29, 30, 33, 37].

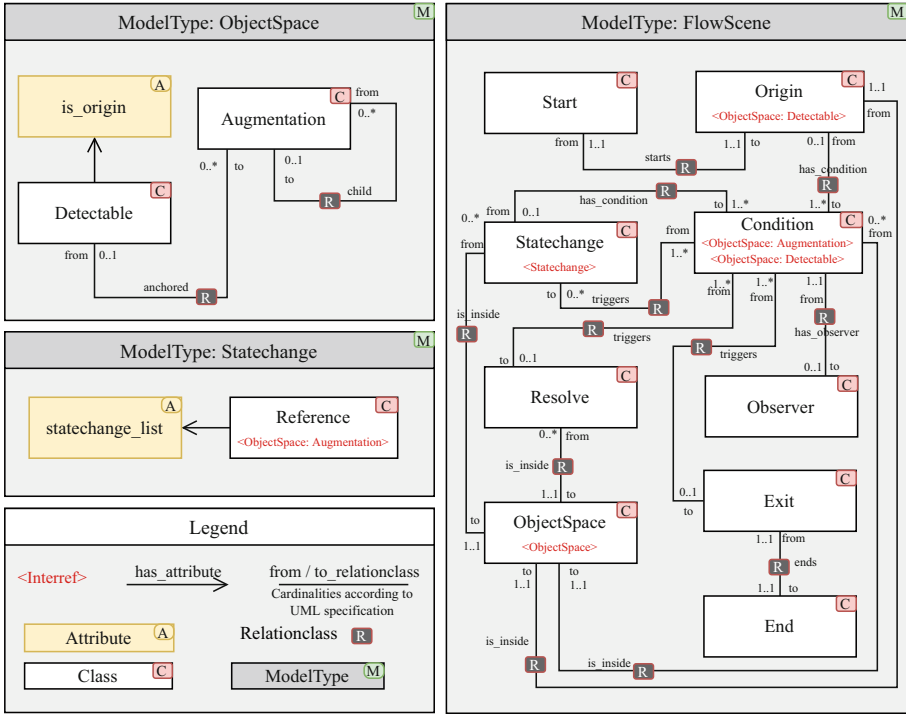
The category *Abstraction* refers to a general aspect for creating an AR modeling language and contains only one specific requirement, which details the generic requirement of different abstraction levels (**GR<sub>5</sub>**). **SR<sub>7</sub>**: To reduce complexity and to separate the different roles 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 [21, 29, 30, 37]. 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<sub>8</sub>**: Due to the nature of modeling languages, an abstract and a concrete syntax in textual notation needs to be provided [13, 20], also for easing future interoperability with previous approaches [6, 15, 21, 29, 30, 33, 37]. In addition, as visual notations are more intuitive and user-friendly than text-based notations, a two-dimensional graphical notation needs to be specified [15]. Finally, since the AR domain reverts largely to 3D content, specifying models directly in a 3D environment is useful to facilitate spatial imagination [6, 33, 37]. Thus, a domain-specific modeling language should consider concepts for text-based, 2D visual, and 3D spatial modeling. **SR<sub>9</sub>**: To allow for an easy and rapid adaptation of the language as requirements change, the modeling language shall be based on *metamodeling* [6, 13, 29]. **SR<sub>10</sub>**: It should be possible to directly feed the model into an AR application for the *execution* of the modeled AR scenario [15, 29, 30, 37]. Thus, a domain-specific modeling language for AR applications shall provide a data format that can be processed by an AR engine during runtime [10] or generate code for creating the AR application itself from the models [15]. **SR<sub>11</sub>**: 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 [18]. **SR<sub>12</sub>**: To ensure reproducibility and accessibility, the implementation of the domain-specific modeling language shall be made openly available [29, 33, 37].

### 4.3 Language Specification

According to Frank the phase of *language specification* contains several parts [13]. 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





**Fig. 2.** Metamodel of the DSML for augmented reality applications with the three modeltypes ObjectSpace, Statechange, and FlowScene, as well as a legend.

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<sup>2</sup>* 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 used the metamodeling language of ADOxx [11]. ADOxx was chosen due to its wide usage within projects of the OMiLAB network [14] and the availability of an open platform for the implementation of model editors. The main metamodeling concepts in ADOxx are [11, 12]: *ModelType* (M), *Class* (C), *Relationclass* (R), and *Attribute* (A). Modeltypes contain one or more classes, which may be connected by relationclasses. Modeltypes, classes, and relationclasses may have attributes. Instances of classes and relationclasses can only be contained in one particular instance of a modeltype. Special attributes of type **<Interref>** act as pointers to other class instances or model instances. In the metamodel introduced in the following, each

concept will be marked with the icons introduced above ((M), (C), (R), (A)) to indicate the corresponding meta<sup>2</sup>-concept.

Figure 2 shows the metamodel of the new domain-specific modeling language. The modeling language is divided into three separate *ModelTypes* (M): *ObjectSpace*, *Statechange*, and *FlowScene*. This results from requirements GR<sub>2</sub>, GR<sub>5</sub> and SR<sub>7</sub>. An *ObjectSpace* (M) defines the real world of an AR environment. It contains the two classes *Augmentation* (C) and *Detectable* (C) as defined by requirements SR<sub>1</sub> and SR<sub>4</sub>. Further, augmentations can include other augmentations, indicated by the *child* (R) relationclass and they may be connected to Detectables via *anchored* (R) relations (SR<sub>3</sub>). A *Detectable* has an attribute *is\_origin* (A), specifying if a *Detectable* references the world origin (SR<sub>2</sub>).

*Statechanges* are described in the separate ModelType *Statechange* (M) - SR<sub>5</sub> and SR<sub>7</sub>. Within such models, Augmentations from the *ObjectSpace* model are referenced (*Reference* (C)) and changes on their attributes - e.g., a rotation transformation - are expressed via the attribute *statechange\_list* (A).





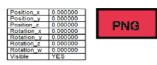














The *FlowScene* (M) ModelType defines the workflow of the AR application and how it reacts to different environmental conditions (GR<sub>4</sub>, SR<sub>6</sub>). Every *FlowScene* contains exactly one *Start* (C) and one *End* (C) instance (SR<sub>6</sub>). Each *FlowScene* contains an *ObjectSpace* (C) instance, which references an instance of the *ObjectSpace* ModelType. Inside this *ObjectSpace* class instance, the *FlowScene* model defines an *Origin* (C), one or multiple *Statechanges* (C), *Conditions* (C), and *Resolves* (C) (SR<sub>2</sub>, SR<sub>6</sub>). They are linked to the *ObjectSpace* with the *is\_inside* (R) relationclass, specifying that these concepts are linked to one specific *ObjectSpace*. The *Origin* is used to define the world origin of the AR environment. Thus, it references a *Detectable* in the *ObjectSpace* model. *Conditions* (C) define requirements which are necessary to trigger the subsequent *Statechanges*, or to trigger *Resolves*, if there are no consecutive *Statechanges* (SR<sub>6</sub>). Thus, *Statechanges* and *Resolves* are connected to *Conditions* by the *triggers* (R) relationclass. *Conditions*, on the other hand, follow an *Origin* or *Statechange* via the *has\_condition* (R) relationclass. Furthermore, *Conditions* can be associated with an *Observer* (C) using the *has\_observer* (R) relationclass. *Observers* can be used to monitor sensor data or APIs (SR<sub>6</sub>).

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 [23]. 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 [4].

#### 4.4 Implementation and Execution

Subsequently, the modeling language has been implemented using the freely available and open ADOxx metamodeling platform and will be made available

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

via Zenodo [25]. The platform allows the easy definition and adaptation of meta-models based on the ADOxx meta<sup>2</sup> model and the creation of model instances in automatically generated model editors (**SR**<sub>9</sub>). ADOxx provides several text-based formats for defining metamodels and models, as well as a DSL for graphical notation (**SR**<sub>8</sub>). 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**<sub>10</sub>). For this purpose, a software component has been designed in the form of an AR engine to interpret the models. The engine is implemented as a platform-independent web application using the 3D JavaScript library *three.js*<sup>15</sup> and the VR/AR immersive web standard *WebXR* [18]. The application can be accessed through a WebXR-compatible web browser on any mobile device, such as smartphones or *head-mounted displays* in line with requirement **SR**<sub>11</sub>. For starting an AR experience, the engine processes the models selected by the user and monitors the user's environment for potentially relevant changes. Based on these environmental changes and user interactions, the application adapts the environment according to the specified workflows specified through triggers, conditions, and actions (**SR**<sub>6</sub>).

## 5 Use Case

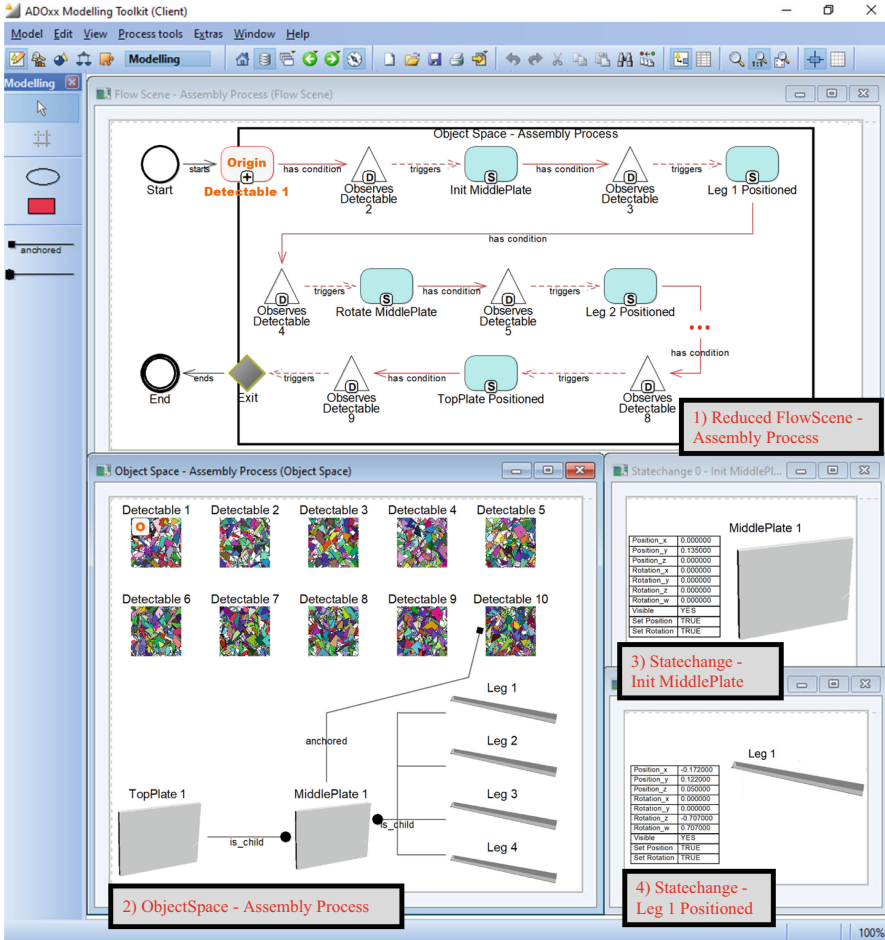
To demonstrate the use of the modeling language and showcase a practical application, we have developed a use case involving augmented reality-assisted assembly of a bedside table. The goal of this use case is to guide a user through the assembly of a bedside table using an augmented reality application instead of traditional 2D instructions on paper. Figure 3 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. 3, 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. 3, 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 furniture piece, e.g., “TopPlate 1”. These *Augmentations* are provided as GLTF files<sup>16</sup>, 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

<sup>15</sup> <https://github.com/mrdoob/three.js/>.

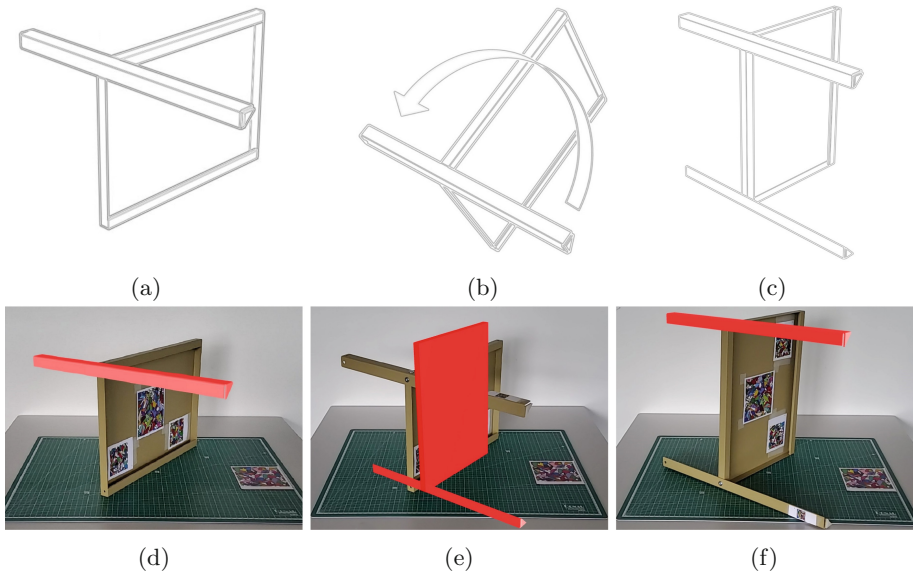
<sup>16</sup> <https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html>.



**Fig. 3.** 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.

*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. 3, 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.



**Fig. 4.** Illustration of the assembly process of a bedside table – cf. IKEA [1] (a–c), and the support through AR based on the visual models (d–f).

The execution of the models of the use case is shown in Fig. 4 by using parts from an IKEA table [1]. Subfigures (a)–(c) illustrate the traditional 2D assembly instructions for (a) “attaching Leg 1”, (b) “turning MiddlePlate 90° counterclockwise”, and (c) “attaching Leg 2”. Subfigures (d)–(f) illustrate the same steps of the instructions in augmented reality using the aforementioned models [25] and the WebXR AR engine. The screenshots were taken while using the WebXR AR engine in the Chrome browser on a *Samsung Galaxy Tab S7* tablet. Subfigure (d) shows the *Statechange* “Leg 1 Positioned”. It superimposes an image of *Leg 1* on top of the real *MiddlePlate*, whose existence, position, and orientation are detected via a marker – *Detectable 10*. The *Statechange* “Rotate MiddlePlate”, where the virtual object is rotated according to the desired position for further assembly of the table is shown in Subfigure (e). Subfigure (f) shows the *Statechange* “Leg 2 Positioned”. The augmentation shows where the next leg shall be attached. As can be seen in subfigures (d), (e) and (f), several colored markers are placed on the real object at strategic points and according to the *ObjectSpace* model. Once a marker is detected, it is decided based on the current state of the workflow defined by the *FlowScene* model if it triggers an action or not. If an action is triggered, the workflow moves on and waits until the next detectable (marker) in line is detected. The flexible structure of the DSML allows multiple workflow paths to be active at the same time by checking for multiple detectables simultaneously. Detectables are also tracked when they are not part of the *FlowScene*. To avoid making the use case unnecessarily complex, the concepts of *Resolves* and *Observer* were not used.

**Table 2.** Feature comparison of the new domain-specific visual modeling language ARWFML based on twelve specific requirements  $SR_{1-12}$ . (Y): Requirement met. (N): Requirement not met. (-): Not specified.

Approach		Ruminski & Walczak 2014	Grambow et al. 2021	Seiger et al. 2021	Lechner 2013	Campos-Lopez et al. 2021	Ruiz-Rube et al. 2020	Wild et al. 2014	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
Implementation	SR8: User Interaction								
	Text-based Modeling	Y	Y	Y	Y	Y	Y	Y	Y
	2D Visual Modeling	N	Y	N	N	N	N	N	Y
	3D Spatial Modeling	N	N	Y	N	Y	N	Y	N
	SR9: Metamodeling	N	N	N	N	Y	Y	N	Y
	SR10: Model Execution	Y	Y	N	-	N	Y	Y	Y
	SR11: Open 3D Standard Support								
	Specification	N	N	N	N	N	N	N	N
	Execution	N	N	N	N	N	N	N	Y
	SR12: Openly Available	N	N	Y	N	N	Y	Y	Y
$\Sigma$ of supported requirements	9	11	11	13	16	18	21	26	

## 6 Evaluation

Several techniques can be chosen to evaluate the new modeling language, including feature comparisons, theoretical and conceptual investigations, and empirical evaluations [34]. Thereby we opted for a feature comparison to previous approaches along the specific requirements that we had formulated. The previous approaches we considered were the ones from Ruminski and Walczak [30],

Grambow et al. [15], Seiger et al. [33], Lechner [21], Campos-Lopez et al. [6], Ruiz-Rube et al. [29], and Wild et al. [37].

For each specific requirement that we had formulated, we conducted a detailed comparison using multiple dimensions, as shown in Table 2. This provides a detailed overview of the features supported by previous approaches and our new modeling language in terms of augmented reality concepts, levels of abstraction, user interaction, metamodeling capabilities, model execution, support for open standards, and availability of according implementations. Thereby, we can show that our new modeling language denoted as *ARWFML* (AR Workflow Modeling Language) currently supports 26 out of 33 dimensions of requirements, whereas the next runner-up only supports 21 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 ( $\mathbf{SR}_6$ ). Concerning *User Interaction* ( $\mathbf{SR}_8$ ), 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 [18] ( $\mathbf{SR}_{11}$ ). This would certainly facilitate the specification of models, as 3D modeling greatly facilitates spatial imagination.

## 7 Conclusion and Outlook

In this paper, we presented a domain-specific visual modeling language that is capable of representing complex augmented reality workflows for diverse application scenarios and that can be executed using the open WebXR standard. The modeling language allows designers to specify three different types of visual models: (1) for defining 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 and allows the user to focus on the content and functionality of AR applications. The technical feasibility was demonstrated by implementing the modeling language using the ADOxx platform and a prototypical web application for executing the models. A first evaluation has been conducted through a feature comparison to previous approaches and indicated the high coverage of the defined requirements.

In future research, we plan a further evaluation of the DSML and the AR application by means of a user study, which allows to identify bottlenecks or blind spots of the DSML. Furthermore, the 2D modeling approach presented here has some limitations 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. Therefore, a new metamodeling platform is currently being developed to incorporate the third dimension during visual modeling, enabling 3D modeling in three-dimensional space [24]. Once the approach has gained further maturity, it will be possible to evaluate it empirically.

## References

1. IKEA Online Shop (2023). <https://www.ikea.com/us/en/p/knarrevik-nightstand-black-30381183/>. Accessed 28 Apr 2023
2. Azuma, R.T.: A survey of augmented reality. *Presence Teleoperators Virtual Environ.* **6**(4), 355–385 (1997)
3. Bock, A.C., Frank, U.: Low-code platform. *Bus. Inf. Syst. Eng.* **63**(6), 733–740 (2021). <https://doi.org/10.1007/s12599-021-00726-8>
4. Bork, D., Roelens, B.: A technique for evaluating and improving the semantic transparency of modeling language notations. *Softw. Syst. Model.* **20**(4), 939–963 (2021). <https://doi.org/10.1007/s10270-021-00895-w>
5. Brunschwig, L., Campos-Lopez, R., Guerra, E., de Lara, J.: Towards domain-specific modelling environments based on augmented reality. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), Madrid, ES, pp. 56–60. IEEE (2021). <https://doi.org/10.1109/ICSE-NIER52604.2021.00020>
6. Campos-López, R., Guerra, E., de Lara, J.: Towards automating the construction of augmented reality interfaces for information systems. In: Insrán, E., et al. (eds.) *Information Systems Development: Crossing Boundaries Between Development and Operations (DevOps) in Information Systems (ISD2021 Proceedings)*, Valencia, Spain, 8–10 September 2021. Universitat Politècnica de València/Association for Information Systems (2021). <https://aisel.aisnet.org/isd2014/proceedings2021/hci/6>
7. Dalton, J., Gillham, J.: Seeing is believing (2019). <https://www.pwc.com/gx/en/industries/technology/publications/economic-impact-of-vr-ar.html>. Accessed 09 Mar 2023
8. Di Ruscio, D., Kolovos, D., de Lara, J., Pierantonio, A., Tisi, M., Wimmer, M.: Low-code development and model-driven engineering: two sides of the same coin? *Softw. Syst. Model.* **21**(2), 437–446 (2022). <https://doi.org/10.1007/s10270-021-00970-2>
9. Doerner, R., Broll, W., Grimm, P., Jung, B. (eds.): *Virtual and Augmented Reality (VR/AR)*. Springer, Cham (2022). <https://doi.org/10.1007/978-3-030-79062-2>
10. Fill, H.-G., Härer, F., Muff, F., Curty, S.: Towards augmented enterprise models as low-code interfaces to digital systems. In: Shishkov, B. (ed.) *BMSD 2021*. LNBIP, vol. 422, pp. 343–352. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-79976-2\\_22](https://doi.org/10.1007/978-3-030-79976-2_22)
11. Fill, H., Karagiannis, D.: On the conceptualisation of modelling methods using the ADOxx meta modelling platform. *Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model.* **8**(1), 4–25 (2013). <https://doi.org/10.18417/emisa.8.1.1>
12. Fill, H.-G., Redmond, T., Karagiannis, D.: Formalizing meta models with FDMM: the ADOxx case. In: Cordeiro, J., Maciaszek, L.A., Filipe, J. (eds.) *ICEIS 2012*. LNBIP, vol. 141, pp. 429–451. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-40654-6\\_26](https://doi.org/10.1007/978-3-642-40654-6_26)

13. Frank, U.: Domain-specific modeling languages: requirements analysis and design guidelines. In: Reinhartz-Berger, I., Sturm, A., Clark, T., Cohen, S., Bettin, J. (eds.) *Domain Engineering, Product Lines, Languages, and Conceptual Models*, pp. 133–157. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-36654-3\\_6](https://doi.org/10.1007/978-3-642-36654-3_6)
14. Götzinger, D., Miron, E.-T., Staffel, F.: OMiLAB: an open collaborative environment for modeling method engineering. In: *Domain-Specific Conceptual Modeling*, pp. 55–76. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-39417-6\\_3](https://doi.org/10.1007/978-3-319-39417-6_3)
15. Grambow, G., Hieber, D., Oberhauser, R., Pogolski, C.: A context and augmented reality BPMN and BPMS extension for industrial Internet of Things processes. In: Marrella, A., Weber, B. (eds.) *BPM 2021. LNBIP*, vol. 436, pp. 379–390. Springer, Cham (2022). [https://doi.org/10.1007/978-3-030-94343-1\\_29](https://doi.org/10.1007/978-3-030-94343-1_29)
16. Gulden, J., Yu, E.: Toward requirements-driven design of visual modeling languages. In: Buchmann, R.A., Karagiannis, D., Kirikova, M. (eds.) *PoEM 2018. LNBIP*, vol. 335, pp. 21–36. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-02302-7\\_2](https://doi.org/10.1007/978-3-030-02302-7_2)
17. Jannaber, S., Riehle, D.M., Delfmann, P., Thomas, O., Becker, J.: Designing a framework for the development of domain-specific process modelling languages. In: Maedche, A., vom Brocke, J., Hevner, A. (eds.) *DESRIST 2017. LNCS*, vol. 10243, pp. 39–54. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-59144-5\\_3](https://doi.org/10.1007/978-3-319-59144-5_3)
18. Jones, B., Goregaokar, M., Cabanier, R.: WebXR device API. W3C candidate recommendation draft, work in progress, World Wide Web Consortium (2023). <https://www.w3.org/TR/2023/CRD-webxr-20230303/>
19. Karagiannis, D., Mayr, H.C., Mylopoulos, J. (eds.): *Domain-Specific Conceptual Modeling, Concepts, Methods and Tools*. Springer, Cham (2016). <https://doi.org/10.1007/978-3-319-39417-6>
20. Karsai, G., Krahn, H., Pinkernell, C., Rumpe, B., Schneider, M., Völkel, S.: Design guidelines for domain specific languages. In: Rossi, M., Sprinkle, J., Gray, J., Tolvanen, J.P. (eds.) *Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM 2009)*, Orlando, vol. B-108, pp. 7–13. Helsingin Kauppakorkeakoulu (2009)
21. Lechner, M.: ARML 2.0 in the context of existing AR data formats. In: Latoschik, M.E., Reiners, D., Blach, R., Figueroa, P.A., Wingrave, C.A. (eds.) *6th Workshop on Software Engineering and Architectures for Realtime Interactive Systems, SEARIS 2013*, Orlando, FL, USA, 17 March 2013, pp. 41–47. IEEE Computer Society (2013). <https://doi.org/10.1109/SEARIS.2013.6798107>
22. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. *ACM Comput. Surv.* **37**(4), 316–344 (2005). <https://doi.org/10.1145/1118890.1118892>
23. Moody, D.L.: The physics of notations: toward a scientific basis for constructing visual notations in software engineering. *IEEE Trans. Softw. Eng.* **35**(6), 756–779 (2009). <https://doi.org/10.1109/TSE.2009.67>
24. Muff, F., Fill, H.: Initial concepts for augmented and virtual reality-based enterprise modeling. In: Lukyanenko, R., Samuel, B.M., Sturm, A. (eds.) *Proceedings of the ER Demos and Posters 2021 Co-Located with 40th International Conference on Conceptual Modeling (ER 2021)*, St. John’s, NL, Canada, 18–21 October 2021. *CEUR Workshop Proceedings*, vol. 2958, pp. 49–54. CEUR-WS.org (2021). <https://ceur-ws.org/Vol-2958/paper9.pdf>
25. Muff, F., Fill, H.: ADOxx Library and UseCase Models for the ER23 Publication: A Domain-Specific Visual Modeling Language for Augmented Reality Applications Using WebXR (2023). <https://doi.org/10.5281/zenodo.8207639>

26. Muff, F., Fill, H.: Past achievements and future opportunities in combining conceptual modeling with VR/AR: a systematic derivation. In: Shishkov, B. (ed.) BMSD 2023. LNBI, vol. 483, pp. 129–144. Springer, Cham (2023). [https://doi.org/10.1007/978-3-031-36757-1\\_8](https://doi.org/10.1007/978-3-031-36757-1_8)
27. Nguyen, T.: 4 impactful technologies from the gartner emerging technologies and trends impact radar for 2021 (2021). <https://www.gartner.com/smarterwithgartner/4-impactful-technologies-from-the-gartner-emerging-technologies-and-trends-impact-radar-for-2021>. Accessed 09 Mar 2023
28. Roo, J.S., Hachet, M.: One reality: augmenting how the physical world is experienced by combining multiple mixed reality modalities. In: Gajos, K., Mankoff, J., Harrison, C. (eds.) Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, UIST 2017, Quebec City, QC, Canada, 22–25 October 2017, pp. 787–795. ACM (2017). <https://doi.org/10.1145/3126594.3126638>
29. Ruiz-Rube, I., Baena-Pérez, R., Mota, J.M., Sánchez, I.A.: Model-driven development of augmented reality-based editors for domain specific languages. *IxD&A* **45**, 246–263 (2020). <https://doi.org/10.55612/s-5002-045-011>
30. Ruminski, D., Walczak, K.: Dynamic composition of interactive AR scenes with the carl language. In: Bourbakis, N.G., Tsihrintzis, G.A., Virvou, M. (eds.) IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications, Chania, Crete, Greece, pp. 329–334. IEEE (2014). <https://doi.org/10.1109/IISA.2014.6878808>
31. Saxena, D., Verma, J.K.: Recreating reality: classification of computer-assisted environments. In: Verma, J.K., Paul, S. (eds.) Advances in Augmented Reality and Virtual Reality. SCI, vol. 998, pp. 3–9. Springer, Singapore (2022). [https://doi.org/10.1007/978-981-16-7220-0\\_1](https://doi.org/10.1007/978-981-16-7220-0_1)
32. Schmalstieg, D.: Augmented Reality: Principles and Practice. Addison-Wesley, Boston (2016)
33. Seiger, R., Kühn, R., Korzetz, M., Aßmann, U.: HoloFlows: modelling of processes for the Internet of Things in mixed reality. *Softw. Syst. Model.* **20**(5), 1465–1489 (2021). <https://doi.org/10.1007/s10270-020-00859-6>
34. Siau, K., Rossi, M.: Evaluation techniques for systems analysis and design modelling methods - a review and comparative analysis. *Inf. Syst. J.* **21**(3), 249–268 (2011). <https://doi.org/10.1111/j.1365-2575.2007.00255.x>
35. Visic, N., Fill, H., Buchmann, R.A., Karagiannis, D.: 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, 13–15 May 2015, pp. 286–297. IEEE (2015). <https://doi.org/10.1109/RCIS.2015.7128889>
36. Wild, F., Perey, C., Hensen, B., Klamma, R.: IEEE standard for augmented reality learning experience models. In: Mitsuhashi, H., et al. (eds.) IEEE International Conference on Teaching, Assessment, and Learning for Engineering, TALE 2020, Takamatsu, Japan, 8–11 December 2020, pp. 1–3. IEEE (2020). <https://doi.org/10.1109/TALE48869.2020.9368405>
37. Wild, F., et al.: Towards data exchange formats for learning experiences in manufacturing workplaces. In: Kravcik, M., Mikroyannidis, A., Pammer, V., Prilla, M., Ullmann, T.D., Wild, F. (eds.) Proceedings of the 4th Workshop on Awareness and Reflection in Technology-Enhanced Learning in conjunction with the 9th European Conference on Technology Enhanced Learning: Open Learning and Teaching in Educational Communities, ARTEL@EC-TEL 2014, Graz, Austria, 16 September 2014. CEUR Workshop Proceedings, vol. 1238, pp. 23–33. CEUR-WS.org (2014). <http://ceur-ws.org/Vol-1238/paper2.pdf>

38. Yin, K., He, Z., Xiong, J., Zou, J., Li, K., Wu, S.T.: Virtual reality and augmented reality displays: advances and future perspectives. *J. Phys. Photonics* **3**(2), 022010 (2021). <https://doi.org/10.1088/2515-7647/abf02e>
39. Zhou, F., Duh, H.B., Billinghurst, M.: 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, 15–18 September 2008, pp. 193–202. IEEE Computer Society (2008). <https://doi.org/10.1109/ISMAR.2008.4637362>