Towards Domain-Specific Modelling Environments based on Augmented Reality

Léa Brunschwig
Univ. Autónoma de Madrid
Madrid, Spain
Lea.Brunschwig@uam.es

Rubén Campos-López
Univ. Autónoma de Madrid
Madrid, Spain
Ruben.Campos@uam.es

Esther Guerra
Univ. Autónoma de Madrid
Madrid, Spain
Esther.Guerra@uam.es

Juan de Lara
Univ. Autónoma de Madrid
Madrid, Spain
Juan.deLara@uam.es

***Abstract*—Models are pervasive in many disciplines, like software and systems engineering. Modelling by the use of domain-specific languages (DSLs) has lowered the entry barrier to this activity to domain experts and citizen developers. At the same time, we are witnessing constant improvements in the capabilities of mobile devices, like augmented reality (AR) based on their camera. AR could be exploited in modelling scenarios that require locating virtual objects in the surrounding physical space.**

In this vision paper, we explore the use of DSLs with AR syntax, and propose the automated synthesis of mobile modelling environments for them. This opens the door to novel scenarios for domain-specific modelling in areas like interior design, Industry 4.0, domotics, tourism and transportation, among others. We propose a design process founded on Software Language Engineering principles, provide prototype tool support, and identify open challenges for the community.

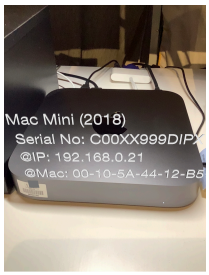
***Index Terms*—Modelling, Domain-Specific Languages, Augmented Reality, Software Language Engineering.**

I. Introduction

Modelling is central to all engineering disciplines to specify, understand, analyse and build the *system under study* (SUS), among other activities. Software engineering is no exception, as models are pervasively used during the development process. Models can be built either using general-purpose modelling languages, like the UML [1], or with domain-specific languages (DSLs) [2], [3] tailored to an application area. Due to their highly specialized syntax, DSLs are sometimes targeted to end-users with no technical background [4].

Traditionally, modelling has been performed either manually in paper or whiteboards, or supported by tools via desktop computers or laptops. More recently, the improvement of the capabilities in mobile devices (like the computer power, touch screen size and components like cameras) has promoted the emergence of environments for modelling on mobile devices [5]–[7]. These environments can be used in mobility, which permits modelling close to the SUS and using the mobile accessories (camera, GPS, gyroscope...). This enables new modelling scenarios which are not possible in desktops, especially when DSLs are designed for non-technical end-users. For example, mobile-enabled DSLs can be used for modelling and monitoring within the premises of a factory; for creating models of touristic or hiking routes on-site; or for modelling the position and configuration of the devices of a smart home while walking through it.

(a) Interior design: Virtual table on a real room.



(b) Inventory: Displaying computer information.

(c) Leisure: Routes and tips about points of interest.

Fig. 1: Examples of AR modelling applications.

This paper goes one step further from current mobile modelling environments by proposing DSLs enhanced with augmented reality (AR) [8] for their use within mobile devices. In our vision, the user will be able to create domain-specific models using the mobile device, but instead of placing the objects on a blank canvas, the camera is used to place the objects (virtually) within the surroundings, using AR. This leads to a new range of modelling applications, like modelling environments for interior design, where furniture (elements of an AR-enabled DSL) can be placed and overlaid over the room the user is in (cf. Fig. 1a); applications for computer inventory, where relevant computer information is displayed on top of the real computer (cf. Fig. 1b); or leisure applications where tourists can display tips on points of interest by focusing the camera on them, and get directions towards nearby touristic spots via arrows on the real-world (cf. Fig. 1c).

AR has become very popular. There are many products to build AR-based applications, ranging from programming libraries like ARKit [9] by Apple or ARCore [10] by Google, to authoring tools like Unity [11]. However, building AR applications from scratch is challenging as it requires deep expertise on the AR programming framework [12]. Instead, we propose the use of Software Language Engineering principles [13], [14] to reduce the implementation effort to create AR-enabled DSLs and avoid the need of coding.

This new ideas paper has the following contributions: (i) we present the concept of AR-based DSL; (ii) we showcase

TABLE I: Categorization of AR-based modelling scenarios. Legend: Physical Object (PO), Virtual Object (VO).

Purpose	Type	Scenario examples	Capabilities
Inform, Augment	D	Inventory; Logistics; Museums; Airports; Train stations	Display static properties of POs Display relations between POs
Monitor	D	Smart factories; Industry 4.0; Digital twins; Networking and data-centres	Display static properties of POs Display relations between POs Display dynamic properties of POs
Control, Configure	D		Display static properties of POs Display relations between POs Display dynamic properties of POs Set property values of POs Create relations between POs
Design	P	Interior and domotic design; Commercial planning; Create hiking/touristic routes	Create VOs Set property values of VOs Create relations between VOs
Animate	P/D	Gaming; Education	Create dynamic VOs Set property values of dynamic VOs

meaningful application scenarios for AR-based DSLs; (iii) we propose a model-based process to define AR-based modelling environments; and (iv) we show a working prototype called ALTER that allows using AR-based DSLs on iOS devices.

II. Motivation and usage scenarios

AR overlays digital elements on the physical world, making them visible by headset devices (like Microsoft's HoloLens [15]) or with the camera displays of mobile devices. This allows users to perceive the real world and the digital objects intertwined. In contrast, in immersive virtual reality (VR) environments, users only perceive the digital world. According to Azuma [8], AR systems combine real and virtual, are interactive in real time, and are registered in 3D.

AR opens the door to new modelling scenarios in domains that can profit from modelling closer to the SUS and its surrounding context. In these scenarios, the model objects can be either virtual surrogates of elements that do not exist in the physical world, or virtual representations of real-world elements enriched with digital information. AR can support both descriptive modelling (i.e., describing an existing system) and prescriptive modelling (i.e., describing a system to be constructed) [16]. In the former case, existing physical objects (e.g., machines in a smart factory) can be augmented with properties stored in the model (e.g., temperature, failure rate). In the latter case, virtual objects (e.g., furniture) can be integrated on the real world (e.g., a room being decorated) and be automatically geolocated based on their position.

Table I classifies AR-modelling scenarios based on their purpose, modelling type (Prescriptive or Descriptive), and modelling capabilities required from the environment.

The first scenario of the table aims at *augmenting* existing physical objects with interesting information for users. An example would be an AR system for a museum, which displays additional data on art works and provides suggestions on museum tours. Modelling in this case is descriptive. To realise the scenario, physical objects may need to be tagged (e.g., via QR codes) to enable its AR augmentation.

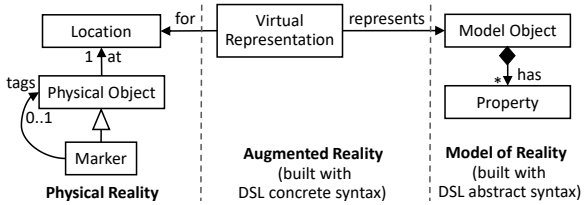


Fig. 2: Augmenting the reality using an AR-based DSL.

The following two scenarios extend the previous one by the ability to display (*monitor*) or modify (*control*, *configure*) information that is dynamic. For example, in smart factories, the modelling environment would overlay dynamic machine data, like failure rates [17]. For this purpose, the model has to be connected with services able to update the displayed information and change it through the AR display. Proposals based on models@runtime [18] might be used as the back-end of this kind of environments.

The fourth scenario supports *designing* some system. An example would be an environment for interior design, to place virtual furniture objects on a real room, in-place (cf. Fig. 1a). Such environments should permit placing AR objects in a physical environment and setting their properties (e.g., weight, material). This modelling scenario is prescriptive since, once created, the model will likely be used to build a real system.

In the last category, the virtual objects placed on the real world may be *animated* and exhibit dynamic behaviour. Examples of this scenario include applications for gaming (e.g., Pokémon-Go style) and education (e.g., an AR visualization of the movements of the solar system planets).

While many tools support developing AR applications, a recent survey [12] providing insights into how AR/VR creators use authoring tools revealed that *they struggled with available design methods and tools, and felt that most required "too much coding"*, and identified the challenge to *support end-user developers as a growing population of AR/VR creators*. Therefore, the next two sections explain our proposal to define AR-based modelling applications using Software Language Engineering principles [13], [14], with no need for coding.

III. Defining augmented reality-based DSLs

In traditional modelling environments, the user interacts with a model of the SUS using a DSL. The model and the SUS are disconnected, in the sense that the model objects are not intertwined with the real objects, and the physical location of the modeller is irrelevant. Instead, our proposal takes the model closer to the SUS by extending the DSL notation with AR, as Fig. 2 shows. In this setting, the physical reality is augmented with model-based data, and the modeller location is used to display the (virtual) model objects positioned nearby.

A DSL is made of an abstract syntax, a concrete syntax and semantics [14]. The abstract syntax describes the primitives of interest, their properties and relations. In model-based approaches, it is defined through a meta-model, commonly a class diagram. The concrete syntax describes how models are

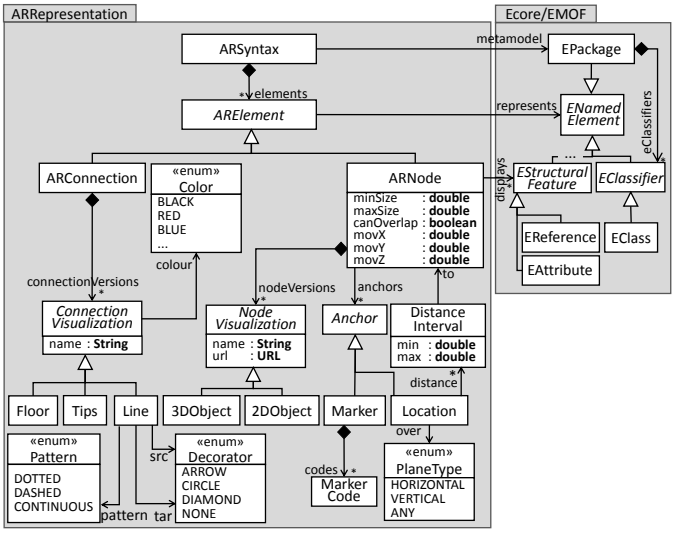


Fig. 3: Meta-model for AR-based concrete syntaxes.

represented, typically leading to graphical or textual DSLs. Finally, the semantics refer to the meaning of the models, and is usually defined by an interpreter, a transformation into another language, or a code generator [14].

The construction of an AR-based DSL requires the definition of an AR-based concrete syntax. We have created the meta-model of Fig. 3 for this purpose. It permits defining AR-based syntaxes by annotating the domain meta-models that describe the abstract syntax. The latter are typically described using technologies like the OMG's meta-object facility (MOF) [19], and popular implementations like the Eclipse Modeling Framework (EMF) [20] (cf. right package in Fig. 3).

Our AR-based syntax distinguishes *nodes* (class ARNode in Fig. 3) and *connections* (class ARConnection). Class ARNode has references to the domain meta-model class it provides a representation for (EClass), and to the set of class properties to be displayed (reference displays). It may define several *visualization* versions (e.g., to display several types of chairs in an interior design application) using either 3DObjects or 2DObjects. Visualizations have a name and are referenced via a URL. They can be built with 3D authoring kits like Blender, or selected from libraries.

Nodes can be placed at suitable *anchors* (class Anchor). Anchors can be markers (e.g., QR codes) or constraints stating admissible locations, such as in a given plane (horizontal, vertical) or at a certain distance interval to another node. For example, a DSL for interior design may require that lamps are located on horizontal planes, within 1 meter to a socket.

Nodes can also define constraints on how users interact with them. Specifically, setting attribute canOverlap to false prevents that two objects are placed one over the other; attributes minSize and maxSize control the increase or decrease in the size of a placed node; and moveX, moveY and moveZ establish the perimeter where a virtual object can be moved once positioned in the world (if zero, the object cannot be moved).

Regarding AR connections, they are visual representations of references between instances of classes of the domain meta-model (class EReference). Similar to nodes, they can have different versions identified by a name and a colour. The meta-model supports three types of connections: Line, Floor and Tip. Lines are displayed as floating lines between two related objects, with decorations in the source and target ends. Floors display a reference as a path to follow in the floor from the source to the target object (cf. Fig 1c). Tips display arrows on the side of the camera steering the position of the target node, mimicking some first-person shooter games.

IV. Deploying Augmented Reality-based DSLs

We propose the architecture in Fig. 4 to define and use AR-based DSLs on mobile devices. It is supported by a prototype tool named AlteR (domAin modeLling using augmenTEd Reality). This is a native iOS app that uses ARKit [9] for creating and handling AR models.

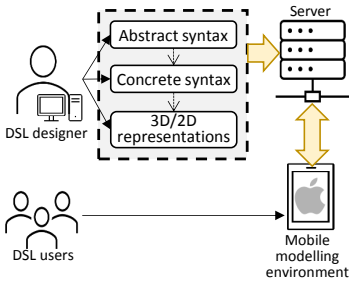


Fig. 4: Architecture of AlteR.

First, to define the AR-based DSL, the DSL designer uses a desktop computer to specify the DSL abstract syntax (domain meta-model) and concrete syntax (an instance of the meta-model in Fig. 3 and 3D/2D object representations), and upload them to a web server. We currently serialize the abstract and concrete syntax in JSON, and the 3D/2D graphics need to follow Apple's SceneKit Scene (SCN) format.

Then, the DSL users access the DSLs available in the server using AlteR on their iPhones/iPads. Upon selecting a DSL, users can start modelling by placing virtual objects (primitives of the DSL) in the real world. Every time a modelling session is started, AlteR/ARKit sets the world origin at the position of the camera of the device. When a virtual object is created, an anchor positioned relative to the established world origin is attached to the object. Models can be saved, which entails taking a snapshot and storing the world origin and the anchors of the model objects. When opening a previous model, the saved anchors are placed at their previous position as soon as the user points the camera to the same location of the snapshot.

Fig. 5 shows a home networking AR-based DSL being used in AlteR. Label 1 is a palette with the items (virtual objects) that can be placed in the real world, named after the classes of the domain meta-model. Label 2 points to a 3D node and depicts the available interactions: *tap* places an item at the tapped location after selecting one from the palette with label 1; *swipe* changes between the visualization versions of a node; *pinch* resizes a node; *long press* a node displays the attributes of the virtual object for their editing, and allows deleting the node; and *pan* moves a node to another position in the world.

Label 3 shows the displayed attributes of a node as a key-value list. Label 4 points at a line connecting two nodes, which represents a reference between two model objects (cf. Section III). Finally, there are two buttons at the top to save

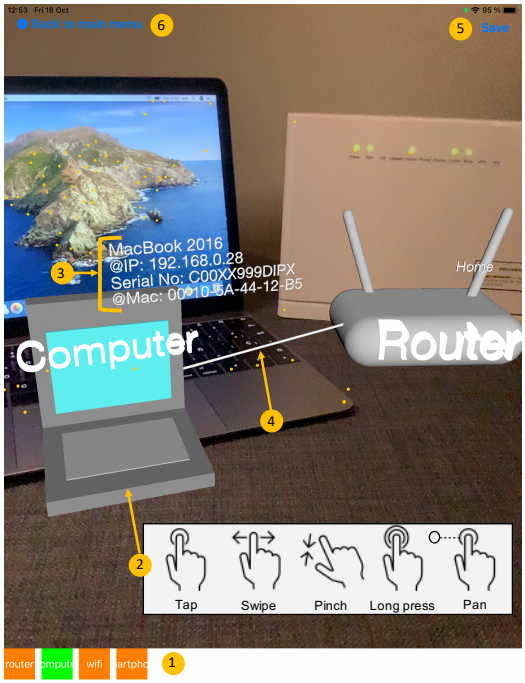


Fig. 5: The main canvas of ALTER (running on an iPad).

the model (label 5) and close the modelling session (label 6). Overall, this DSL enables users quick access to the whole network configuration, without checking in every device.

V. Related work

The recent capabilities of mobile devices have enabled their use for modelling. For example, FlexiSketch [7] and CEL [21] support conceptual modelling using the touch interface of tablets. Other works benefit from advanced features of mobile devices. For example, *active DSLs* [6] support geolocation and external interaction for their use in mobility. Our proposal goes further by providing an AR syntax to DSLs.

AR-based DSLs are displayed in 3D. In this respect, Devil3D [22] can generate environments for 3D visual modelling languages, but it runs on desktops and does not support AR. 3D representations combined with VR has been used in Software Engineering, e.g., to visualize metrics [23].

The advances in mobile devices and the availability of hardware for head-mounted displays (like HoloLens) have prompted the proposal of applications that use AR. Examples include emergency situations [24] (visualising geolocated data reported during an emergency), quality assurance in industrial settings [25] (to display extra information over vehicle components), to help exploring exhibitions [26] (by displaying an AR map towards different parts of the exhibition), or for factory maintenance (to monitor dynamic machine parameters [17], or to configure IoT devices [27]). These applications were built ad-hoc using manual programming. However, as several researchers point out, creating such apps using current technologies is difficult [12], [28]. Our proposal is a step towards alleviating this problem for certain kinds of AR applications.

There are many AR tools and platforms (e.g., DART [29], ARtoolkit [30]). Some of them (AR.js [31], Argon4 [32]) follow a web development style enriched with AR concepts, resulting in apps that run on special browsers independently of the device. Some libraries are device-specific (e.g., ARKit for iOS devices), while others are multi-platform (e.g., ARCore). Beyond programming libraries, platforms like Unity AR Foundations [11], MAXST AR SDK [33], Wikitude AR SDK [34] or Vuforia Studio [35] offer dedicated environments for building multi-platform AR apps. Finally, some approaches, like ProtoAR [36], target the creation of 3D content and its virtual placement over markers. While these are all generic tools to create AR apps, our proposal enables the creation of AR-based DSLs without resorting to coding.

VI. Conclusions and open challenges

In this paper, we proposed the concept of AR-based DSL and the use of Software Language Engineering for defining AR-based modelling environments. Our prototype ALTER illustrates the viability of our ideas. We are currently working on some capabilities listed in Table I which ALTER does not fully support, like QR code anchors for monitoring and control scenarios. We also plan to provide a cloud environment to define the abstract and concrete syntax of AR-based DSLs.

We have argued on the potential of AR-based DSLs to tackle some scenarios, but many challenges remain to fully realize this vision. First, modelling is inherently collaborative, and AR-based modelling is no exception. Hence, ways to specify and realize collaboration are worth exploring [37]. Second, the usability of DSLs is very important, especially in unusual domains. While some heuristics for the design of cognitively effective visual notations have been proposed [38], these may need to be adapted for AR-based DSLs. Another aspect related to usability is interaction, i.e., mapping gestures to editing actions. Our prototype has explored camera-based AR and screen-based interaction (cf. Fig. 5). However, interaction via body gestures (e.g., with the hands) are necessary when using head-set devices such as glasses, contributing to blur the distinction between the virtual and physical worlds, and making AR-based modelling more natural. Likewise, we have proposed object anchoring based on markers or positional constraints, but other trackers based on advanced image recognition (e.g., hands, faces) could be interesting for some DSLs (e.g., in fashion). Finally, our approach is no-code, but friendly ways to define domain meta-models and AR syntaxes by end-users are required to make the approach accessible to them.

VII. Data availability

Further details and videos of our tool ALTER are available at https://alter-ar.github.io.

Acknowledgment

Work funded by EU Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement n° 813884, Spanish Ministry of Science (RTI2018-095255-B-I00) and R&D programme of Madrid (P2018/TCS-4314).

REFERENCES

[1] OMG, "UML 2.5.1 OMG specification," http://www.omg.org/spec/UML/2.5.1/, 2017.

[2] S. Kelly and J. Tolvanen, *Domain-specific modeling - Enabling full code generation*. Wiley, 2008.

[3] M. Voelter, S. Benz, C. Dietrich, B. Engelmann, M. Helander, L. C. L. Kats, E. Visser, and G. Wachsmuth, *DSL Engineering - Designing, Implementing and Using Domain-Specific Languages*. dslbook.org, 2013.

[4] M. Mernik, J. Heering, and A. M. Sloane, "When and how to develop domain-specific languages," *ACM Comput. Surv.*, vol. 37, no. 4, pp. 316–344, 2005.

[5] A. Sebastian-Lombraña, E. Guerra, and J. de Lara, "Positioning-based domain-specific modelling through mobile devices," in *EUROMICRO conference on Software Engineering and Advanced Applications (SEAA'2020)*, ser. IEEE Computer Society, 2020, pp. 150–157.

[6] D. Vaquero-Melchor, J. Palomares, E. Guerra, and J. de Lara, "Active domain-specific languages: Making every mobile user a modeller," in *20th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS*. IEEE Computer Society, 2017, pp. 75–82.

[7] D. Wüest, N. Seyff, and M. Glinz, "Flexisketch: a lightweight sketching and metamodeling approach for end-users," *Softw. Syst. Model.*, vol. 18, no. 2, pp. 1513–1541, 2019.

[8] R. T. Azuma, "A survey of augmented reality," *Presence Teleoperators Virtual Environ.*, vol. 6, no. 4, pp. 355–385, 1997.

[9] ARKit, https://developer.apple.com/augmented-reality/, 2020.

[10] ARCore, https://developers.google.com/ar, 2020.

[11] Unity, https://unity.com/unity/features/ar, 2020.

[12] N. Ashtari, A. Bunt, J. McGrenere, M. Nebeling, and P. K. Chilana, "Creating augmented and virtual reality applications: Current practices, challenges, and opportunities," in *CHI '20: CHI Conference on Human Factors in Computing Systems*. ACM, 2020, pp. 1–13.

[13] R. Lämmel, *Software Languages: Syntax, Semantics, and Metaprogramming*. Springer, 2018.

[14] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice, Second Edition*, ser. Synthesis Lectures on Software Engineering. San Rafael, California (USA): Morgan & Claypool Publishers, 2017.

[15] HoloLens, https://www.microsoft.com/en-us/hololens, 2020.

[16] R. Salay, "Using modeler intent in software engineering," Ph.D. dissertation, Department of Computer Science, University of Toronto, 2010.

[17] S. Coscetti, D. Moroni, G. Pieri, and M. Tampucci, "Factory maintenance application using augmented reality," in *APPIS 2020: 3rd International Conference on Applications of Intelligent Systems*. ACM, 2020, pp. 22:1–22:6.

[18] G. S. Blair, N. Bencomo, and R. B. France, "Models@run.time," *IEEE Computer*, vol. 42, no. 10, pp. 22–27, 2009.

[19] MOF, http://www.omg.org/spec/MOF, 2016.

[26] N. Hube, M. Müller, and R. Groh, "Facilitating exploration on exhibitions with augmented reality," in *Proc. AVI*. ACM, 2018, pp. 64:1–64:3.

[20] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework, 2nd Edition*. Upper Saddle River, NJ: Addison-Wesley Professional, 2008.

[21] R. Lemma, M. Lanza, and A. Mocci, "CEL: touching software modeling in essence," in *22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering, SANER*. IEEE Computer Society, 2015, pp. 439–448.

[22] J. Wolter, "Devil3d - A generator framework for three-dimensional visual languages," in *Proceedings of the 18th International Conference on Distributed Multimedia Systems, DMS*. Knowledge Systems Institute, 2012, pp. 171–176.

[23] S. Romano, N. Capece, U. Erra, G. Scanniello, and M. Lanza, "On the use of virtual reality in software visualization: The case of the city metaphor," *Inf. Softw. Technol.*, vol. 114, pp. 92–106, 2019.

[24] A. Campos, N. Correia, T. Romão, I. L. Nunes, and M. Simões-Marques, "Mobile augmented reality techniques for emergency response," in *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*. ACM, 2019, pp. 31–39.

[25] N. Hube, M. Müller, J. Wojdziak, F. Hannß, and R. Groh, "Towards augmented reality in quality assurance processes," in *Proceedings MMVE@MMSys*. ACM, 2018, pp. 16–21.

[27] R. Seiger, M. Gohlke, and U. Aßmann, "Augmented reality-based process modelling for the internet of things with holoflows," in *Enterprise, Business-Process and Information Systems Modeling - 20th International Conference, BPMDS*, ser. Lecture Notes in Business Information Processing, vol. 352. Springer, 2019, pp. 115–129.

[28] M. Nebeling and M. Speicher, "The trouble with augmented reality/virtual reality authoring tools," in *IEEE International Symposium on Mixed and Augmented Reality, ISMAR*. IEEE, 2018, pp. 333–337.

[29] B. MacIntyre, M. Gandy, S. Dow, and J. D. Bolter, "DART: a toolkit for rapid design exploration of augmented reality experiences," *ACM Trans. Graph.*, vol. 24, no. 3, p. 932, 2005.

[30] H. Kato and M. Billinghurst, "Developing AR applications with artoolkit," in *3rd IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2004, p. 305.

[31] AR.js, https://ar-js-org.github.io/AR.js-Docs/, 2020.

[32] Argon4, https://app.argonjs.io, 2020.

[33] MAXST AR SDK, http://maxst.com#/en/arsdk, 2020.

[34] Wikitude AR SDK, https://www.wikitude.com, 2020.

[35] Vuforia Studio, https://www.ptc.com/en/products/vuforia, 2020.

[36] M. Nebeling, J. Nebeling, A. Yu, and R. Rumble, "Protoar: Rapid physical-digital prototyping of mobile augmented reality applications," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI*. ACM, 2018, p. 353.

[37] M. Billinghurst and H. Kato, "Collaborative augmented reality," *Commun. ACM*, vol. 45, no. 7, pp. 64–70, 2002.

[38] D. L. Moody, "The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering," *IEEE Trans. Software Eng.*, vol. 35, no. 6, pp. 756–779, 2009.