A Data Management Approach for Modular Industrial Augmented Reality Applications

Jan Luca Siewert[1[0000-0002-1115-7594]], Matthias Neges[1[0000-0002-3424-3620]], and Detlef Gerhard[1[0000-0002-3266-7526]]

[1] Ruhr-University Bochum, Germany
{Jan.Siewert,Matthias.Neges,Detlef.Gerhard}@ruhr-uni-bochum.de

Abstract. Modular architectures for Industrial Augmented Reality (IAR) applications allow more flexibility and can lower the barrier of entry, compared to custom-built applications dominant today. This contribution presents a data management approach for such architectures. Work plans are converted into a presentation-independent format that can be consumed by the system. Based on available modules and capabilities, related data, like CAD models, images, or descriptions, are automatically converted into a format suitable for the modules responsible for presenting the content. This can include converting a CAD module into a tessellated format, but also generating a rendering in cases where 3D models cannot be display, e.g., in a projection-based AR experience.

In IAR applications, the position of the displayed content in the users surrounding is an important aspect. For industrial contexts, this mostly relates to presenting content at specific parts of larger assemblies. This contribution shows how different combination of existing AR tracking technologies, in addition to CAD data of such assemblies, can be used to setup flexible IAR systems.

Keywords: Industrial Augmented Reality, Data Management, Product Lifecycle Management.

1 Introduction

With growing maturity of available Augmented Reality devices, data management approaches become increasingly important. Industrial Augmented Reality (IAR) has various usecases across the whole product lifecycle [1]. Research has shown advantages of using AR: during service tasks workers are reliable guided through unknown procedures [2]; precise spatial information can be transmitted for remote maintenance on complex equipment [3]; process reliability is increased in quality control applications. Nonetheless, IAR applications have not found widespread adoption in the industry, yet [4]. Today's IAR applications mostly fall in one of two main categories: they are custom-built using a Software Development Kit or are provided as-a-service for a specific application. While the first approach results in highly customized systems tailored to the specific need for the specific use-cases, both their initial creation and later customization requires highly trained professionals, making this approach only suitable for larger companies. As-a-service products mostly cover single applications,

e.g., maintenance or remote support IAR systems that are flexible enough that they can be adopted and expanded upon by smaller companies as well need to be composed of modular, reusable, and expandable parts.

This contribution presents a novel data management approach for such a modular IAR architecture. Instead of creating a monolithic IAR application from scratch or trying to adapt an as-a-service product, this architecture proposes the use of stand-alone modules, that can be composed, adapted, and reconfigured to fit a specific use-case. After the architecture is introduced, it is shown hot it can adapt to a specific assembly use-case.

2 Prior Works

While various studies reported on various advantages of IAR systems, there remain obstacles before a widespread use can be achieved. IAR systems should be integrated into existing data management infrastructure, like PLM and IoT systems. To achieve that, these systems need to communicate using standardized interfaces and data formats [5]. Because there is no "one-size-fits-all" solutions to support all tasks, both the content and the application logic needs to be adapted to fit a given use-case. Today, this often requires expert knowledge, making the process time-consuming and expansive [6]. To implement IAR systems cost-effective and with minimal set-up time, an expandable and adaptable approach needs to be taken [7, 8]. To achieve expandable and flexible IAR systems, various modular IAR architectures have been proposed.

MacWilliams et al. identified a set of common components in AR applications. These include an application subsystem, containing the actual, use-case dependent application logic. The interface to the user consists of interaction subsystems on the one hand and presentation subsystems on the other hand. Tracking subsystems are responsible to display the content at a fixed position in space. All data is part of a world subsystems, while user specific content is part of a context subsystem [9].

Kuster et al. propose a service-based architecture for IAR applications. Services register their capabilities at a central service registry. Exemplary services are extracting payloads from QR codes or asking for input from a user. These services are then composed to business process using the BPMN graphical modelling language. The actual display and interaction concepts used by the various AR devices used, however, is not specified [10].

To support assembly tasks with IAR applications, numerous approaches for an (semi-) automatic generation of work instructions based on CAD assembly documents exist. Neb et al. extract high-level "assembly features" from form features using a macro in a CAD system. These are then the basis for an automatic generation of an assembly sequence as well as additional information, like assembly distance and a suitable animation to use in a visualization. The resulting data is exported in a structured format together with a tessellated geometry. Combining these, an AR Head-Mounted-Display (HMD) is then used to visualize the assembly instruction [11].

Gors et al. show that a fully automatic content generation approach is not suitable for real-world CAD models, because elastic parts, like springs or cables, are only added

as static entities in the assembly model. Here, candidates for the next assembly step are found by identifying parts that are unobscured by the already assembled parts of the structure. However, manual intervention is necessary by an operator when the algorithms cannot find a next part because of the stated limitations. After an assembly sequence is identified, instruction content is created. This includes text describing the part and the required assembly direction, image sequences, 3D models of the various intermediate assembly steps, and animations [12].

3 Concept

The proposed modular architecture is roughly based upon a classic 3-layer architecture, while incorporating the most relevant components of AR systems as discussed by MacWilliams et al. [9]. The view layer can be divided into multiple interaction and presentation systems. These can be developed independently from any special use-case, allowing them to be reused in different settings. An application logic module reacts to events from the interaction systems and forwards content to suitable presentation systems.

3.1 System Architecture

The overall system architecture (Figure 1) is roughly based on a classic 3-layer architecture. The view layer consists of the afore mentioned presentation and interaction systems. They communicate with a runtime through a message broker. The runtime is responsible for executing the actual application logic. It receives events from the interaction systems and reacts by sending data to be presented by the presentation systems. This is based upon the data prepared by the planning module. This module takes the source data from the data layer and brings it in a format that is understood by the runtime. Thereby it considers which interaction and presentation systems are available, as well as the requirements of the use-case. This process is presented in more detail in the next section.

On the data layer, three modules for data preparation are responsible for creating the content based on input data, convert the data to suitable formats, and act as a data proxy to simplify access. Data source can include different systems, like PLM for CAD data and assembly information, ERP systems for work plans, or IoT data for telemetry. The necessary adapters are implemented as small services. All available services are registered at a central service registry.

Content creation can happen manually by an operator, for example in a stand-alone web-application or as part of a broader ERP system. It relates the input data, like CAD models or assemblies, to the specific work plan that will be executed by the worker.

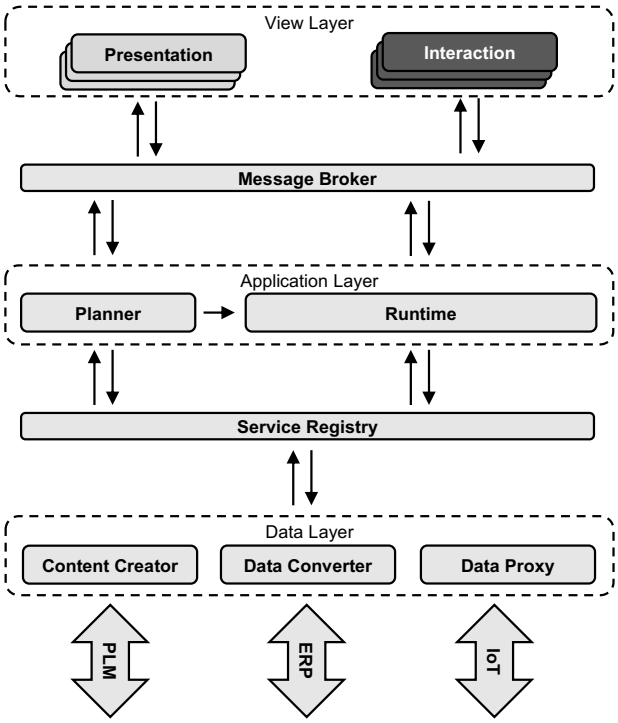

Fig. 1. Overall System Architecture

Data conversion includes converting the same data type into different format, e.g., different formats for images. It can also mean transforming the data in a new format. For example, a 3D model can be transformed into an image by creating a thumbnail.

Sometimes, a direct connection between an AR device and other software system might be undesirable. For example, a device in a production shopfloor should not have direct access to the supporting PLM system. In such a case, the data can be transferred through a data proxy, which also supports caching data and forwarding new data back to the source systems.

The details of the presentation and interaction systems are shown in the Block diagram in Figure 2. An interaction system is described by the different event types it supports. A presentation system is described by both the data it can represent, like images, or 3D models, and the anchor points it can attach data to.

3.2 Implementing a Use-Case

To meet changing requirements from different users or different use-cases, the assistant system must be composed from the different available subsystems and their capabilities. Most of the required steps can run automatically. Some, however, may require input from an operator who is responsible for maintaining the system. These are shown as manual actions in the BPMN diagram in Figure 3.

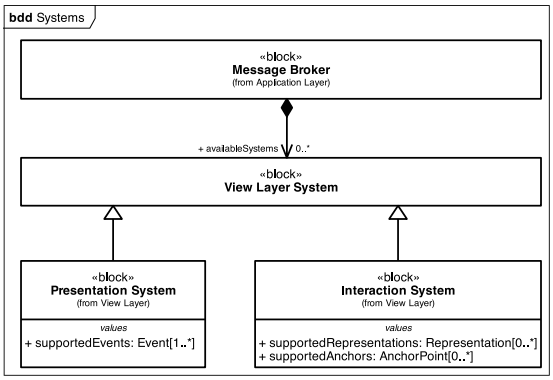

Fig. 2. Block Diagram with the requirements of the Presentation and Interaction Systems.

Work Plan Creation. In a first step, the work instruction is created. Depending on the source data, this step can be performed automatically, the operator can create them manually, or he is assisted in the creation. After this step, a standardized description of the work to be performed is created. The work is divided into individual steps that are connected through transitions. Each transition is tied to a specific event. For example, one event might always start the next step in a list of steps. For every step, one or more pieces of information are displayed at a given anchor. The same information can be conveyed to the worker in different ways, like with an animation, a static 3D model, or a combination of a description and a rendering. A set of minimal system requirements in the form of required events, anchors and minimum adequate representations is derived from the work plan.

Not every IAR application needs a concrete, step-by-step work plan. However, the architecture requires a standardized definition of the displayed data to derivate the required anchor points and content representations. For those use-cases, the "work plan" associates available data-points with their real-world location and implies

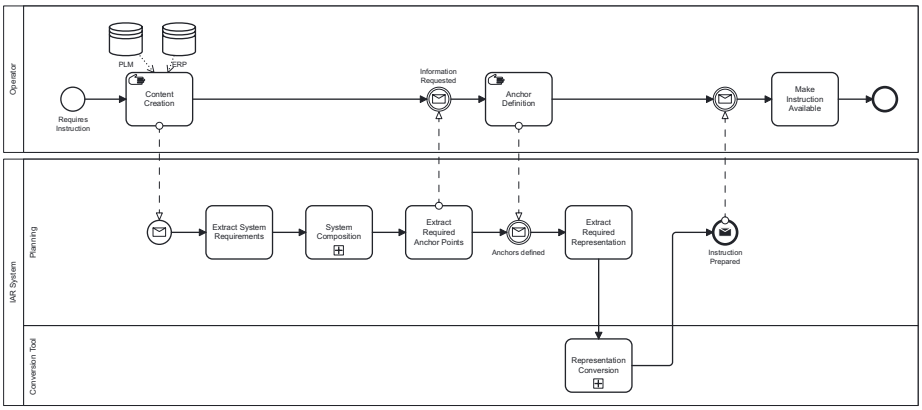

Fig. 3. Process for creating content for an IAR system in the proposed architecture.

arbitrary transitions between each entry. This enables a worker to freely select displayed content.

System Composition. In the next step, subsystems are selected to match these requirements. Because all subsystems register themselves with their capabilities at a central registry. From there, the optimal set of subsystems is selected to support the use case.

Anchor Definition. More so than the display of three-dimensional content, an AR application is characterized by being able of presenting content at a specific point in space. This point remains static even when the user moves and is typically referred to as an anchor. While content might be shown at an arbitrary position in space, IAR applications mostly present information tied to a specific position at a machine or an individual part in it. A presentation system might support one or more of anchor types, like model targets or image markers. Some might be capable of supporting any anchor of a given type, e.g., any QR code independently of its actual payload. Mostly, supporting a specific anchor requires additional work that must be performed automatically or manually by the operator. For model-based tracking algorithms, some preprocessing is necessary based on either available CAD models, or by scanning a physical part before the worker can start his work. In some cases, the operator might assign static position to a given anchor. An example might be an assembly station where work pieces are always stored at the same location. Here, no special anchors are needed to be capable of highlighting those.

Representation Conversion. A system usually has to support required anchors to convey the necessary information to the worker. A system, however, is more flexible with the specific representations it uses therein. When the required anchors are provided by the overall system, different data representations are automatically created from the source data. The presentation subsystems describe their capabilities. This can include the type of representation, like 3D models, images, or texts, as well as supported file formats. The resulting representations are permanently saved so that there are available later on.

Representations can be created by converting different file formats of the same type, e.g., creating a tessellated OBJ file from a step file, as well as by transforming then, e.g., derivate a 2D rendering from a CAD file. These conversion and transformations are done by stand-alone services that register their capabilities at the registry. They are then tasked on demand to create the representations to best fit the requirements and content of the use-case.

System Expansion. Creating an IAR application for a given use-case with the proposed architecture does encourage modularity and reusability. Individual subsystems, like converters, but also interaction and representation systems, are implemented only once. When their functionality is required for another use-case, they work out of the box. To support other use-cases with different requirements, the system needs to be adapted.

The work plan conversion is strongly tied to both the specific use-case as well as the source data. For different use-cases, this module is therefore different. It is, however, indifferent to the actual output to the user. The type of output devices, like a projection-

based AR setup or AR glasses, the work plan creation process stays the same. The work plan creation has to transform the source data into discrete states with representations attached to each and transitions between them.

Representation and interaction systems are independent from the actual use-case. When a use-case requires special interactions, like sending a measured value in a quality assurance process, a new event type has to be introduced. Then, an existing system gets expanded with the functionality to emit these events. These events should be defined as general as possible to be able to reuse them in other contexts as well.

When the requirements of a use-case cannot be fulfilled even after attempting to convert representations, additional systems need be created and registered. This can be new representation systems that support new anchors or representation types, or additional converters or transformers that support different output types. In this case, the operator gets a list of unfulfillable requirements as well as suggestions on how to solve them.

4 Implementation

To demonstrate the system, several prototypes have been implemented and tested. To show how different presentation systems can convey the same information, an assembly use-case is assisted by an in-situ projection system on the one hand. On the other hand, a worker has access to the same information in a HoloLens-based HMD system. Because IAR systems based on the proposed architecture are easily adaptable, the HMD system is later expanded to support a simple Quality Control (QC) use case as well. The implemented modules and the data flow between them are shown in Figure 4.

Both systems are controlled through a runtime module written on-top of Node-JS, while Eclipse Mosquitto and the MQTT protocol were chosen for the Message Broker functionality. A Fusion Data Adapter module written in Python connects to Autodesk Fusin 360 Manage as a PLM system and Autodesk Platform Services to perform data conversions. It makes the methods available as a REST-API.

4.1 Supporting an Assembly Use-Case

Work Plan Creation. The input data is already in a structured format. The individual assembly steps are converted to states. Each state has the textual description associated to it. Where available, each step also has a 2D rendering of the current assembly state. Between steps, transitions are introduced corresponding to the events “next” and “previous”.

System Composition. Each subsystem describes its capabilities using a defined JSON structure. For interaction systems, these include the types and names of the events it supports. For presentation systems, the capability includes the type and file formats of supported representations. Furthermore, they supply a list of anchors that are supported without manual work by the operator. Based on this information, the subsystems suitable for assisting the specific applications are selected.

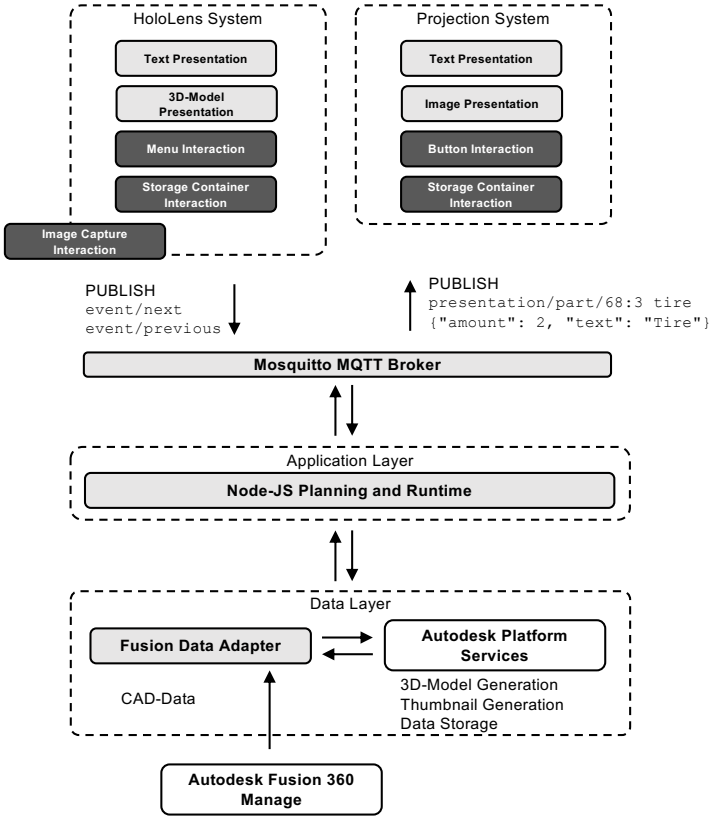

Fig. 4. Implemented modules and data flow between them.

For the projection system, presentation modules to display text and images have been implemented, respectively. They are only able to present information without any anchors. For interaction, the setup can project buttons on the workbench. Through a depth sensor mounted next to the projector, the users can interact with these by hovering their hand above them. Similarly, the system can highlight storage containers attached to the workbench and recognize when the user takes a part out of these.

The HoloLens has similar functionalities. Instead of images, it can render complete and interactive 3D models. The simple button functionalities are supported through a more sophisticated menu that is attached to the user's hand.

Anchor Definition. Each specific anchor point has a type and a unique identifier. For anchor points describing parts, this should be same identifier as in the Autodesk Fusion 360 Manage PLM system. Additionally, a system can declare a list of anchor types it might support. When a use-case requires an anchor that is not already supported but with a type that might be, the system gets request indicating the specific anchor to support.

How the subsystem handles such a request depends on its implementation. In this implementation, the operator setting up the system must create all anchors manually by defining the storage containers. Both the projection setup and the HMD make suggestions for potential storage containers based on their depth map and environment sensing capabilities, respectively. Then, the operator associates the container area with the specific part by selecting it from a dropdown menu with the previously undefined parts.

When a new anchor is supported by a subsystem, it updates its registered capabilities by updating the retained message at the MQTT broker. This restarts the planning process.

Representation Conversion. After all anchor requirements are fulfilled, the runtime converts data into representations supported by the presentation systems. Through the Autodesk Platform Services Platform, various types of 3D-CAD data formats can be converted into tessellated descriptions of the geometry based on the OBJ file format. A Fusion Data Adapter module uses this API to convert the CAD data and to store the results for subsequent executions. Additionally, the CAD data can be described by creating thumbnail images for the various models.

4.2 Supporting a Quality Control Use-Case

To showcase the **System Expansion** step of the proposes architecture, the HoloLens application is expanded with an additional interaction system to ask the user for numerical inputs. Furthermore, the user can capture images and annotate them to document issues. These support an additional event that sends the data gathered from the user to the runtime. Source data for this Quality Control use-case is again described as a state machine with new transitions where user input is required. Because additional

Fig. 5. The projection (left) and HMD system for assembly (center). The worker can take and annotate images for quality control in the QC application. in use.

modules can subscribe to these events on the MQTT broker, a new reporting module stores inputted data outside of the architecture and writes it into a quality control report.

Figure 5 shows the implemented systems in use. First the projection setup for assembly of robot models, then the HoloLens application supporting the same use-case, and finally the additional quality control use-case on the HoloLens.

5 Conclusion

The proposed modular IAR architecture enables a more flexible approach to create IAR applications. By using encapsulated modules, the system is easily expandable to more use cases. This contribution presents an approach for data management and preparation that takes into account the specific requirements for a given use-case as well as the capabilities of the different modules that are available in a given setting. By enabling operators to easily configure their systems for new use-cases, adaption can be quicker and more cost-effective. Through automatic conversion of different source data, the systems easily integrate into existing processes and data management software.

The architecture imposes a certain overhead, both during setup and during execution. Therefore, it might not be suitable for bigger companies that want to create highly specialized assistant systems to a lot of workers. At the same time, the architecture must be further tested to validate the possible repurposing and adaption of an existing system to a completely different use case.

In future works, the proposed service-based architecture could be the foundation for new business models for AR modules. Because the individual modules are stand-alone and reusable for a right range of use-cases, software vendors might offer specific subsystems, e.g., for model tracking, that can then be added to the specific IAR system on demand. One example could be the use of Artificial Intelligence to automatically detect errors during an assembly task. This could be added to the system as a new type of interaction subsystem that emits specific events, like going to the next step because the assembly is correct or showing an error message.

References

1. Pierre Fite-Georgel. Is there a reality in industrial augmented reality? International Symposium on Mixed and Augmented Reality, Science and Technology Proceedings. IEEE, 2011.
2. Mario Lorenz, Sebastian Knopp, Philipp Klimant. Industrial Augmented Reality: Requirements for an augmented reality maintenance worker support system. Adjunct Proceedings of the 2018 IEEE International Symposium on Mixed and Augmented Reality (Ismar), 151–153, 2018.
3. Riccardo Masoni, Francesco Ferrise, Monica Bordegoni, Michele Gattullo, Antonio E. Uva, Michele Fiorentino, Ernesto Carrabba, Michele Di Donato. Supporting Remote Maintenance in Industry 4.0 through Augmented Reality. Procedia Manufacturing, 11:1296–1302, 2017.
4. Alberto Martinetti, Henrique Costa Marques, Sarbjeet Singh, Leo van Dongen. Reflections on the Limited Pervasiveness of Augmented Reality in Industrial Sectors. Applied Sciences, 9(16):3382, 2019.

5. Johannes Egger, Tariq Masood. Augmented Reality in Support of Intelligent Manufacturing – A systematic Literature Review. Computers & Industrial Engineering, 140:106195, 2020.
6. Tariq Masood, Johannes Egger. Augmented Reality in Support of Industry 4.0 – Implemetation challenges and success factors. Robotics and Computer-Integrated Manufacturing, 58:181–195, 2019.
7. Moritz Quandt, Benjamin Knoke, Christian Gorldt, Michael Freitag, Klaus-Dieter Thoben. General Requirements for Industrial Augmented Reality Applications. Procedia CIRP, 72:1130–1135, 2018
8. Luís Fernando de Souza Cardoso, Flávia Cristina Martins Queiroz Mariano, Ezequiel Roberto Zorzal. A survey of industrial augmented reality. Computers & Industrial Engineering, 139:106159, 2019.
9. Asa MacWilliams, Thomas Reicher, Gudrun Klinker, Bernd Bruegge. Design Patterns for Augmented Reality Systems. Proceedings of the International Workshop Exploring the Design and Engineering of Mixed Reality Systems (MIXER), Funchal, Madeira, CEUR Workshop Proceedings, Stycze 2004.
10. Tobias Kuster, Nils Masuch, Johannes Fahndrich, Gudrun Tschirner-Vinke, Jan Taschner, Markus Specker, Hendrik Iben, Hannes Baumann, Falko Schmid, Jorg Stocklein, et al. A Distributed Architecture for Modular and Dynamic Augmented Reality Processes. 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), Jul 2019.
11. Alexander Neb, David Brandt, Greg Rauhöft, Ramez Awad, Johannes Scholz, Thomas Bauernhansl. A novel approach to generate augmented reality assembly assistance automatically from cad models. Procedia CIRP, 104:68–73, 2021. 54th CIRP CMS 2021 - Towards Digitalized Manufacturing 4.0.Johannes Egger, Tariq Masood. Augmented reality in support of intelligent manufacturing – a systematic literature review. Computers & Industrial Engineering, 140:106195, Feb 2020.
12. Dorothy Gors, Jeroen Put, Bram Vanherle, Maarten Witters, Kris Luyten. Semi-automatic extraction of digital work instructions from cad models. Procedia CIRP, 97:39–44, 2021. 8th CIRP Conference of Assembly Technology and Systems.