frontiers | Frontiers in Virtual Reality

TYPE Original Research
PUBLISHED 13 January 2023
DOI 10.3389/frvir.2023.1019080



OPEN ACCESS

EDITED BY
Jack Shen-Kuen Chang,
National Cheng Kung University, Taiwan

REVIEWED BY
Jiawei Huang,
Environmental Systems Research Institute, United States
Weiya Chen,
Huazhong University of Science and Technology, China

*CORRESPONDENCE
Xinyi Tu,
xinyi.tu@aalto.fi

SPECIALTY SECTION
This article was submitted to Technologies for VR, a section of the journal Frontiers in Virtual Reality

RECEIVED 14 August 2022
ACCEPTED 05 January 2023
PUBLISHED 13 January 2023

CITATION
Tu X, Autiosalo J, Ala-Laurinaho R, Yang C, Salminen P and Tammi K (2023), TwinXR: Method for using digital twin descriptions in industrial eXtended reality applications. *Front. Virtual Real.* 4:1019080. doi: 10.3389/frvir.2023.1019080



# TwinXR: Method for using digital twin descriptions in industrial eXtended reality applications

Xinyi Tu*, Juuso Autiosalo, Riku Ala-Laurinaho, Chao Yang, Pauli Salminen and Kari Tammi

Mechatronics group, Mechanical Engineering, School of Engineering, Aalto University, Espoo, Finland

Digital twins (DTs) and eXtended Reality (XR) are two core technological enablers for engineering in the Metaverse that can accelerate the human-centric Industry 5.0 transformation. The digital twin technology provides a digital representation of a physical asset with data linkages for inspection, monitoring, and prediction of complex processes or systems, while eXtended reality offers real-and-virtual combined environments for human users to interact with machines. However, the synergies between digital twins and eXtended reality remain understudied. This work addresses this research gap by introducing a novel method "TwinXR" that leverages ontology-based descriptions of Digital twins, i.e., digital twin documents, in industrial eXtended reality applications. To ease the use of the TwinXR method, we publish a Unity package that allows data flow and conversion between eXtended reality applications and digital twin documents on the server. Finally, the work applies the TwinXR method in two industrial eXtended reality applications involving overhead cranes and a robot arm to demonstrate the use and indicate the validity of the method. We conclude that the TwinXR method is a promising way to advance the synergies between digital twins and eXtended reality: For eXtended reality, TwinXR enables efficient and scalable eXtended reality development; For digital twins, TwinXR unlocks and demonstrates the potential of digital twins for data interchange and system interoperation. Future work includes introducing more detailed principles of Semantic Web and Knowledge Graph, as well as developing factory-level TwinXR-compatible applications.

KEYWORDS

**eXtended reality, digital twins, metaverse, ontology, metadata, cyber manufacturing, human-machine interaction**

## 1 Introduction

The fourth industrial revolution (Industry 4.0) originated in 2011 (Vogel-Heuser and Hess, 2016), and has since become a globally-adopted term and focus of significant research, with Smart Factories (Zuehlke, 2010) among the key initiatives. The next industrial revolution (Industry 5.0), meanwhile, recognizes the power of industry to achieve societal goals and to support the long-term service of humanity (Breque et al., 2021), with the arrival of the "Age of Augmentation" (Longo et al., 2020). Numerous promising technologies and applications are expected to accelerate Industry 5.0 transformation. Among those, the Metaverse, a concept proposed in 1992, which has recently also become a popular paradigm in public (Sparkes, 2021), can play a critical role in Industry 5.0 by bringing stunningly immersive digital landscapes to enrich the human experience on factory floors. Industry 5.0 towards the Metaverse features two core technologies, Digital Twins (DTs) and eXtended Reality (XR), which empower humans with access to critical insight and control over diverse machines, systems, and processes (Lee

et al., 2021). The DT technology provides a digital representation of its physical counterpart with data linkages between the two, which enables inspection, monitoring, and prediction of complex processes or systems (Autiosalo et al., 2020), while XR, including Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR), offers real-and-virtual combined environments for human users to interact with machines (Chuah, 2018).

We identified the significance of combining the DT and XR technologies. A user interface is a critical component for any DT system to bring its value to and get inputs from human users. However, the increasing amount of DT data within different formats and from different resources creates a challenging context for the user interface design (Zhu et al., 2019). XR technologies have the exact potential to enable informative and intuitive interfaces. Meanwhile, DTs bring with them a rich amount of data, which serves as the foundation for creating digital content in XR interfaces (Ma et al., 2019). Industrial solutions combining the technologies of DT and XR have been proposed in several research works: Begout et al. (2022) implemented a DT in the context of a reconfigurable factory with the help of an AR authoring tool; Podder et al. (2022) leveraged a DT and its models in VR to enhance the learning experience and productivity of energy efficiency construction workers; Weistroffer et al. (2022) proposed a framework to evaluate a cobotic workstation, by simulating a physics-based DT and using XR to display the DT and its associated data. Besides, our previous research explored the DT-XR combination by creating XR interfaces for a DT-based overhead crane: Tu et al. (2021) implemented a Mixed Reality interface for on-site crane operation, while Yang et al. (2022) proposed an XR development framework with crane virtual training and remote monitoring use cases. However, these works only focused on proof-of-concept technical design and implementation of industrial XR applications for DT systems. Systematic methods are still missing to achieve the synergies between DTs and XR, which are critical in the context of Industry 5.0 towards the Metaverse.

As a primary step to approach XR-DT synergies, common knowledge representation among XR interfaces and machine DTs seems to be necessary. Ontologies have long been considered as a framework for representing and managing shareable and reusable knowledge across domains (Brewster and O'Hara, 2004), and are therefore used in our work to link DTs and XR. In particular, we leverage ontology-based descriptions shared among knowledge-based XR interfaces and information management-oriented DTs.

Górski et al. (2019) presented an approach for knowledge formalization and management in industrial VR applications: Replacing the traditional paradigms of closed VR development with hard-coded knowledge, the proposed open solution could record the product/process knowledge in an application formally and store it outside the application for easy access later. Flotyński (2020) proposed knowledge-based explorable XR environments to open new opportunities for knowledge exploration with the capabilities of monitoring, analyzing, comprehending, examining, and controlling XR environments, as well as users' or assets' behaviors. In addition, it reviewed the existing XR development solutions that were limited to 3D representation purposes, and suggested employing ontology approaches in XR scene creation. Flotyński (2022) suggested that semantically configurable XR environments would unlock the possibility to modify XR application states, and launch the environment from an altered state.

Information management-oriented DTs focus on semantic connections and information flow among different assets and linked applications (Liu et al., 2021). Several machine-readable ontologies can be used for DT communication, such as Schema.org (2022), SAREF (2021), and GS1, (2021), which are typically formatted in JSON-LD (Sporny et al., 2020) or other Linked Data formats. In this article, we refer to the general concept of a document describing a DT with common data ontology as a "DT document." Several specifications for DT documents have been created, including Digital Twin Definition Language (Microsoft, 2020), Web of Things Thing Description (Kaebisch et al., 2020), and Asset Administration Shell (Plattform Industrie 4.0, 2020). These DT document specifications, among others, are compared in Jacoby and Usländer (2020). For development efficiency, this work leverages the one that the authors previously drafted in Ala-Laurinaho et al. (2020). Nevertheless, the TwinXR method is generic and compatible with different specifications. With DT documents describing different DTs and relations among them, we can expect the emergence of a global network of interlinked DTs, the "Digital Twin Web" (DTW) (Autiosalo et al., 2021). Resembling the World Wide Web (WWW), the DTW is expected to have similar standards of open availability, connectivity, and extendability. The Feature-based Digital Twin Framework (Autiosalo et al., 2020) formed a conceptual foundation for the DTW, followed by the design practices specified in Ala-Laurinaho et al. (2020). Based on these, the authors introduced a platform "Twinbase" (Autiosalo et al., 2021), the first implementation of a DTW server, for managing and distributing DT documents, which is leveraged in the implementation of this work. Mattila et al. (2022) demonstrates the use of DT documents and Twinbase in controlling a Smart Factory.

The previous works of the authors can be summarized into two tracks: One was focused on industrial XR application development for DT systems, which is limited to technical design and implementation of XR applications for certain use cases; The other track was focused on creating, connecting, and managing DTs, which can be a powerful tool for information management among machines and connected applications, yet has not been employed in the XR application development practices. A systematic approach to combining XR and DTs to advance the synergy is still missing. With existing solutions, XR application development is a laborious process: One XR application is often tailored to one specific purpose and environment. On-demand composition and modification of XR applications require using specific XR development tools, which are rarely accomplished only by domain experts who oversee the requirements of XR applications from certain machines and their operating environments. Furthermore, data mapping between the physical layer of machines and the XR application layer is often done in an instance-by-instance tedious manner. This work proposes the TwinXR approach that systematically combines information-management DTs and knowledge-based XR. Following the architecture and workflows of the proposed approach, a TwinXR-compatible application can be created once and centrally, then efficiently scaled to new instances according to new machine or environment specifications by simply modifying the associated DT descriptions, without repeating the XR development step. DT descriptions serve as an intermediate layer connecting machines and XR applications, ensuring data interchange and system interoperation. The TwinXR approach enables building generic XR solutions for industrial purposes involving multiple types of users.

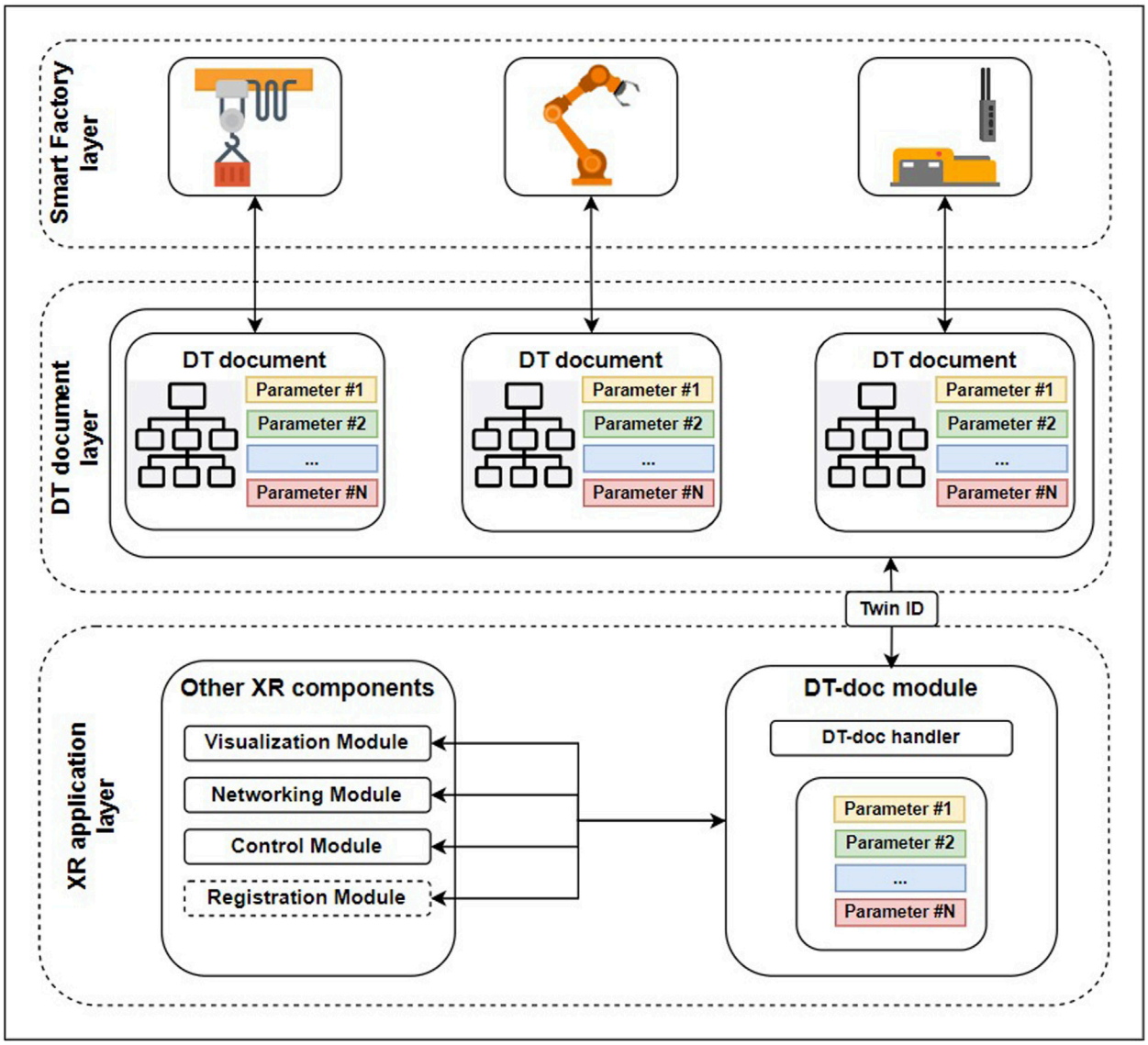


FIGURE 1
Overall architecture of the TwinXR method setup, with three layers from Smart Factory, DT document, to XR application.

The main contribution of this work lies in advancing the synergies between DTs and XR through proposing a novel method TwinXR that leverages DT descriptions in industrial XR applications. The contribution can be further detailed in the following three points.

- Introduce the TwinXR method, which uses descriptions of DTs of Smart Factory devices in creating and instantiating their XR applications.
- Develop a publicly available Unity package (Tu, 2022b) to ease TwinXR-compatible application development.
- Apply the TwinXR method in two industrial XR use cases to demonstrate the usage and validity of the method.

As a known limitation, our current TwinXR method implementation does not use any specific ontology tools, which remain to be developed in the future to enable the widespread adoption of the method.

# 2 Materials and methods

This chapter starts with introducing a three-layered architecture of the TwinXR method, with the materials for each layer provided. Next, we present the workflows of TwinXR-compatible application development, including their creation and instantiation. Finally, the work provides step-by-step processes of users' initializing applications, as well as the two scenarios of an XR application reading and modifying a DT document.

## 2.1 TwinXR setup

This section describes the architecture and the materials used for implementing the TwinXR method. As illustrated in Figure 1, the architecture consists of three layers, from the Smart Factory layer, through the DT document layer, to the XR application layer, with

bi-directional data flow between them. This section covers the general options for hardware or software setup, as well as the specific ones used in the implementation of Chapter 3.

### 2.1.1 Smart factory layer

The Smart Factory layer contains physical machine instances, connected with DTs and equipped with XR interfaces. The machines, such as cranes, robot arms, and mobile robots are diverse, but their XR applications can still contain similar parameters, such as the name of the device, weight of the load, and location in relation to factory coordinates. In the TwinXR method, the metadata of these machines, including the customizable parameters of their XR interfaces, are described in their corresponding DT documents on the server. In the future, whole machine-specific XR application modules could be fetched *via* addresses in the DT documents.

The implementation of TwinXR-compatible applications, described in Chapter 3, involves two use cases, one with overhead cranes, and the other with a robot arm. We use the industrial overhead crane named "Ilmatar" at the Aalto Industrial Internet Campus as a reference model for the crane use case. "Ilmatar" is a Konecranes CXT family crane with Siemens PLCs and can lift 3.2-ton loads along three dimensions using its subsystems of the bridge, trolley, and hoist. "Ilmatar" functions as a DT-based platform for students and researchers to conduct experiments on Smart Factory-related use cases and are equipped with various external digital solutions, including XR interfaces (Tu et al., 2021; Yang et al., 2022) for user interaction, and an OPC Unified Architecture (UA) server for data access. The metadata of such cranes includes the range of their operationally safe zone, the location for their target positioning control, as well as the basic information on their equipped external applications. For the robot arm use case, we use the Universal Robots e-Series robot, UR5e, as a reference model. UR5e robot arm is equipped with six joints and a wide scope of flexibility. It can handle end effectors and fixtures, process and transfer products. The robot arm is typically connected to and controlled with external software such as Robot Operating System (ROS). The metadata of such robot arms includes the speed range, orientation range, and offset of each joint, elbow, and end effector.

### 2.1.2 DT document layer

The DT document layer contains various DT documents hosted on one or more servers. As a middle layer, the DT document layer is connected to the Smart Factory and XR application layers with the bi-directional data flow. Different DT documents are one-to-one mapped with different machine instances. Meanwhile, multiple DT documents can share one XR application, in which case these DT documents would have the same data structure of their shared XR-related parameters, so that the XR application can interchangeably fetch parameters from different DT documents in the same way. The shared parameters from different DT documents are likely of different values, determined by their associated machine instances and operating environments. These parameter values with DT features then flow to the XR application and are linked to other XR components, which eventually customize an instantiated application.

In practice, this work leverages the DT document standard developed in the authors' previous work Ala-Laurinaho et al. (2020). Under this standard, each DT document describes the metadata and features of a single DT in the format of YAML and JSON. A DT document contains mandatory fields, such as name, identifier, and description of the DT, as well as optional fields, such as manufacturer, location of the physical product, and connected services. The document is designed to function along with a Data Link that connects the DT features behind a single access point.

This work further extends the fields in DT documents to include the identified customizable XR features. The XR-related fields are categorized into design and control parameter types: Design parameters are typically those related to XR components' appearance, such as the color, shape, size, opacity, and style, while control parameters are those critical for machine operation, such as the target location, subsystems' moving range, operational zone, and marker location. This work provides two examples of the *descriptions of terms* for the DT document of the overhead crane and the robot arm with XR-related fields, which is publicly available in the GitHub repository (Tu and Autiosalo, 2022). Chapter 3 will elaborate on it along with the TwinXR-compatible application implementation. The data that is passed from the DT document layer to the XR application layer includes both customizable XR features and selected general DT information. The former is used to customize XR applications to fulfill both design and control specifications, while the latter includes Twin ID which is used for the mapping between external applications and DT documents, and other general DT information that is visualized and displayed in XR applications.

Regarding the server implementation, the work leverages Twinbase (Autiosalo et al., 2021), a git-based open-source platform, for managing and distributing DT documents. Twinbase development is hosted by Aalto University and initiated as a result of multiple DT-related projects. Twinbase leverages free-of-charge GitHub services, including GitHub repository, GitHub Actions, and GitHub Pages. New Twinbase server instances can be created by copying the template repository and DT documents can be added with a provided template. A static website hosted by GitHub Pages functions as the primary human user interface of Twinbase, which allows browsing of Twinbase or a selected twin, as well as directs users to a correct GitHub website for creating or modifying a Twinbase or a twin. Chapter 3 will show a Twin page of the "Ilmatar" crane with XR application features. Each DT document contains its Twin ID that is used for referring to the twin and can redirect to the hosting URL of that document. With a Twin ID, machine users or external applications can fetch the corresponding hosting URL and DT document, which is supported by a Python client library (Autiosalo et al., 2021). To further adapt the solution to common XR development environments, this work develops a Unity package containing a C# script *JsonReader.cs* of similar functionality for accessing the Twinbase server. The source code of the package is publicly available in the GitHub repository (Tu, 2022b).

### 2.1.3 XR application layer

The XR application layer consists of a DT-doc module and common XR components. The DT-doc handler in the DT-doc module enables bi-directional data flow and data format conversion between DT documents on the server and their local copy in the XR application. A Twin ID input is required to determine the mapping from an XR application toward its targeted DT document. Common XR components, such as visualization, networking, control, and registration modules, contain customizable parameters that correspond to the ones in the local copy of the DT document. The values of customizable XR parameters can flow in both directions between the DT document and other XR

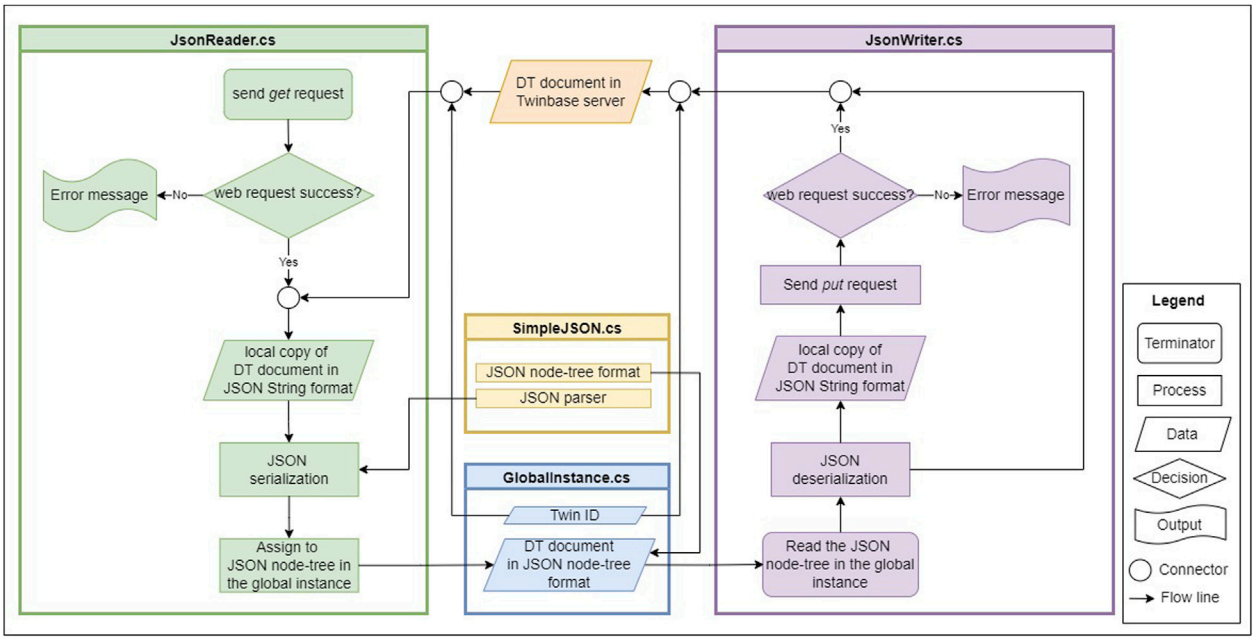


FIGURE 2
Technical desciption of the TwinXR package consisting of four scripts, namely, *GlobalInstance.cs*, *JsonReader.cs*, *JsonWriter.cs*, and *SimpleJSON.cs*.

modules: Other modules can read data from the DT document to determine certain customizable features, while users can modify the DT document through XR interactions. This article uses the term "TwinXR-compatible" to describe an XR application of such thus being compatible with the TwinXR method.

Common XR components of this layer can be developed leveraging existing solutions, including 3D formats like X3D, programming languages like Java and C#, libraries like OpenGL and WebGL, 3D modeling tools like Blender and 3ds Max, and game engines like Unity and Unreal Engine. The established applications can run on various XR devices, such as Meta Quest 2 (formerly Oculus Quest 2) for VR, Microsoft HoloLens 2, and Varjo XR-3 for MR, and mobile devices with Android or iOS operating systems for AR. Among these options for software and hardware setup, this work utilizes the development platform Unity and Trimble XR10 with HoloLens 2 edition for the implementation in Chapter 3.

This work creates an open-source Unity package publicly available in Tu (2022b) to ease the DT-doc module establishment. As shown in Figure 2, the package contains four C# scripts, which together enable bi-directional data flow and conversion between a DT document on a Twinbase server and its local copy in an XR application:

- *GlobalInstance.cs* that creates a global instance living across different scenes of an XR application to store the Twin ID and the local copy of a DT document so that the parameters can be easily accessed by other XR components;
- *JsonReader.cs* that fetches a DT document (in JSON String format) from a Twinbase server *via* a *get* request, converts the DT document into a local copy (in JSON node-tree format), and assigns it to the global instance in the XR application;
- *JsonWriter.cs* that reads the local copy of the DT document (in JSON node-tree format) from the global instance, converts it into JSON String format, and uses it to update the DT document on the Twinbase server *via* a *put* request;
- *SimpleJSON.cs* (Göbel, 2022) that defines the JSON node-tree format and provides the function of a JSON parser, which are used by the other scripts *GlobalInstance.cs*, and *JsonReader.cs*.

## 2.2 Developing and using TwinXR-compatible applications

This section presents the workflows of developing and using TwinXR-compatible applications. The development includes the creation and instantiation of TwinXR-compatible applications, conducted by developers. Once an application is established, users can then initialize it and follow the two workflows of reading and modifying DT documents with XR applications.

Figure 3 presents the workflows for using the TwinXR method for XR application development, from creating the first TwinXR-compatible application instance, i.e., the *origin*, to adapting it into new *instances*, with a focus on the transition and linkages between the two workflows. The creation workflow is conducted once and centrally, involving actions both for a DT document server *origin*, and an XR application *origin*. Leveraging the outcomes of the creation workflow, new developers can directly modify DT documents on their own server *instances* according to the specifications from new machines or operating environments, and consequently, have their own XR application *instances* with customized features.

The creation of TwinXR-compatible application follows the process below: First, customizable XR parameters are identified from the existing XR application; Then, a DT document that includes the identified parameters is created; Meanwhile, a DT-doc module is added to the XR application; Finally, the XR application is modified to connect XR features with parameters from the DT

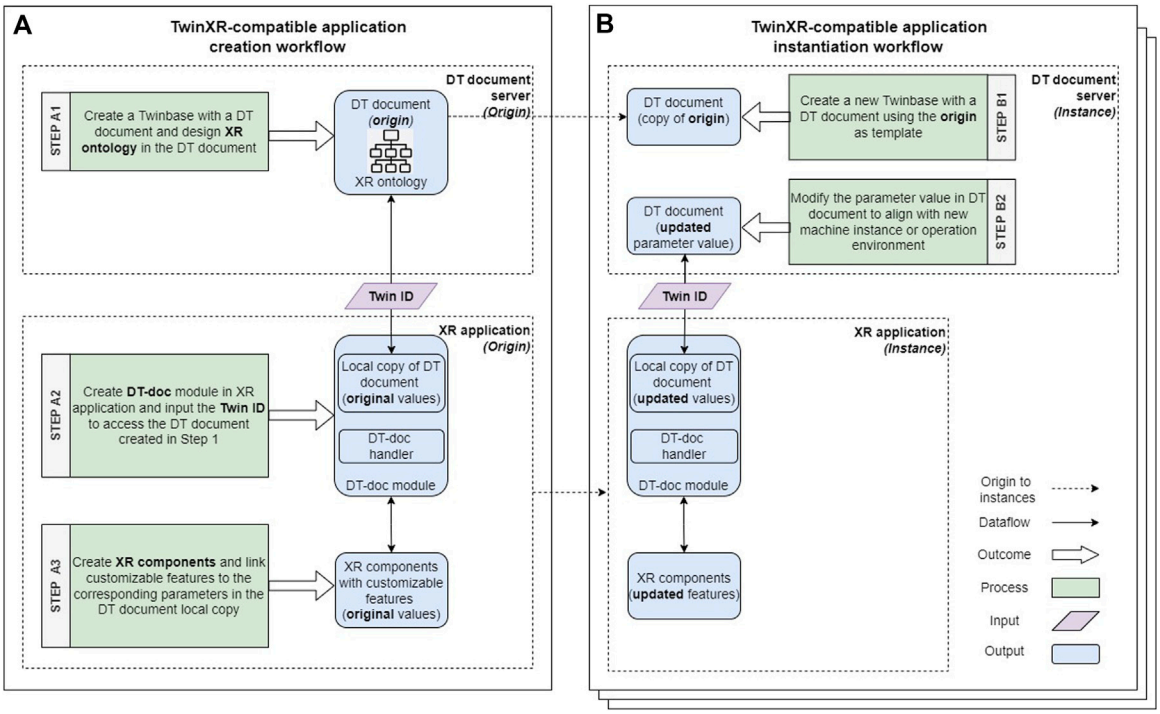


FIGURE 3
Workflows of creating a TwinXR-compatible application *origin* and adapting it to new *instances*. **(A)** TwinXR-compatible application creation workflow. **(B)** TwinXR-compatible application instantiation workflow.

document. Any TwinXR-compatible application *origin* can then be instantiated for a new machine instance. It is only required to create a DT document that contains values of the instance-specific parameters.

To initialize the TwinXR-compatible application, users input the Twin ID stored in the DT document of the targeted machine, whereby the application is linked to the previously established DT document. Once the TwinXR-compatible application is initialized, users can follow the workflows below to read and modify a DT document: Once the application is launched, users first input the Twin ID, through which, the DT-doc module fetches the corresponding DT document; Then, other XR modules read parameters from the fetched local copy of the DT document, which is then used to customize the application; Hereby users can start to use the customized application. Meanwhile, users can modify the local copy of the DT document through XR interactions; Then, the DT-doc module sends the modified local DT document to the server and confirms that the DT document on the server has been modified successfully.

## 3 Results

This chapter demonstrates the use of the TwinXR method in industrial applications with proof-of-concept implementation for operating typical devices in a Smart Factory setup, including overhead cranes and a robot arm. Following the workflows presented in the previous chapter, the work first defines the XR ontology in the DT documents, then implements a Twinbase server with the DT documents, as well as TwinXR-compatible XR applications. The source code of the implementation is publicly available in the GitHub repositories of the Twinbase (Tu and Autiosalo, 2022) and the Unity MR project (Tu, 2022a).

We use the industrial overhead crane named "Ilmatar" at the Aalto Industrial Internet Campus as a reference model for the crane use case, and the Universal Robots e-Series robot, UR5e, as a reference model for the robot arm use case. Figure 4 depicts the architecture of the TwinXR method implementation for crane operation. The robot arm case follows a similar architecture of the TwinXR method implementation to the crane case. The implementation consists of two parts: A Twinbase server with the DT documents, as well as an MR application running on the Trimble XR10 with HoloLens 2 device.

A scenario of using the crane or robot arm application goes as follows: First, users would scan either the QR code of the Twin ID of the "Ilmatar" crane, the demo crane, or the robot arm, and press "*Read DT doc from server*" virtual button; This step triggers the DT-doc module to send a *get* request to the URL redirection services, which then redirect the Twin ID to the corresponding DT document on Twinbase; Consequently, the DT-module fetches the DT document and converts it to a local copy; Then, the visualization and control modules read parameters from the local DT document; Accordingly, the application is customized in terms of, e.g., the content DT document dashboard, the coverage of the safety zone indicator, the location of the target hologram (for the crane case), the speed range, orientation range, and offset of each joint (for the robot arm case), etc. For the crane case, users can also move the target hologram and press the "*Write to DT doc to server*" virtual button to confirm; This step leads to an update on the local DT document with the current target location; The DT-doc module sends the modified local DT document to replace the one on Twinbase. The steps above are repeatable while

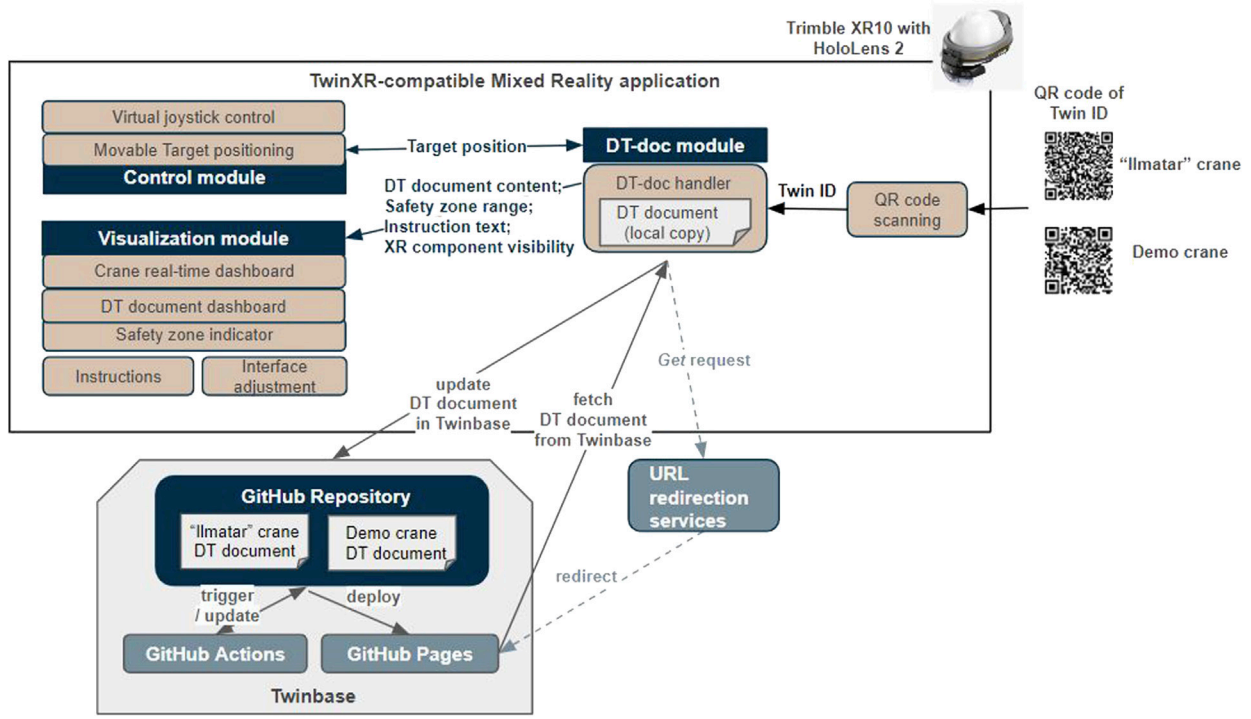


FIGURE 4
Architecture of the TwinXR method implementation for crane operation with a TwinXR-compatible MR application, and a Twinbase server with crane DT documents.

using the application. In other terms, users can scan the QR code of another Twin ID, and fetch or modify the DT document on the Twinbase server anytime according to need.

## 3.1 XR ontology in DT documents

The work first identifies the customizable XR features and defines the XR ontology in the DT documents. A full list of the *descriptions of terms* in the DT document, including both XR ontology and general DT terms, can be found in the GitHub repository (Tu and Autiosalo, 2022).

Tables 1, 2 list the XR-related terms with their types, descriptions, and data formats in the crane and robot arm DT documents respectively. We categorize the terms into design and control types: The design terms determine the user interface (UI) and experience (UX) related features, while the control terms determine operation-critical features. The design terms include the color, shape, size, angle, scale, style or visibility of XR components like the dashboard, target, and safety zone indicator, as well as the instruction about using the application. There is often a need to tailor these terms according to the specifications of machines, operating environments, and processes for user-friendliness and safety reasons. The control terms include the location of a marker for spatial registration that enables an XR application to pose holograms in relation to the physical world, and the range of a safety zone indicator that outlines the operationally safe area for the crane or robot arm movement. For the crane case with target positioning for crane movement, the control terms also include the location of a target; For the robot arm case, the control terms include the offset orientation, speed range, and orientation range of the six robot joints. As the marker location, safety zone range, target location, offset orientation, speed range, and orientation of each joint often vary in different operating environments, updating them through DT documents eases the instantiation of a new XR application.

## 3.2 Twinbase with DT documents

The work implements the Twinbase server with the provided template and therefore shares the same components that were elaborated by Autiosalo et al. (2021): a GitHub repository, GitHub Actions, and GitHub Pages, as illustrated in Figure 4. The GitHub repository contains DT documents and other files, which are updated by the GitHub Actions and deployed to the GitHub Pages for distribution. For Twin IDs, this work uses the URL redirection service at dtid.org, hosted by Rebrandly and introduced by Autiosalo et al. (2021). The Twin IDs are redirected to the corresponding twins on the GitHub Pages. As a special measure due to a bug in the *url* property in *UnityWebRequest* class, the Twin IDs used in this article also have an accompanying redirection toward the JSON version of the DT document. The implemented Twinbase server is then used to host DT documents for both crane and robot arm cases.

Based on the defined XR ontology in Table 1, we create two crane DT documents on Twinbase. One is for the "Ilmatar" crane, which corresponds to a DT document *origin* as the one in Figure 3 described in the previous chapter. The other is for a demo crane of the same type

**TABLE 1** *Descriptions of terms* **for XR ontology in the crane DT documents.**

| Category | Term | Description | Format |
|---|---|---|---|
| Design | targetColor | The color of the target hologram, by default yellow | String |
| Design | targetShape | The shape of the target hologram, by default sphere | String |
| Design | targetSize | The size of the target hologram (cm), by default 20 (diameter) | Float |
| Design | targetOpacity | The opacity of the target holograms (%), by default 70 | Float |
| Design | dashboardPosition | The position of the dashboard center with regard to the user (m), by default (1, 0, 0) | Array of three floats |
| Design | dashboardScale | The scale of the dashboard, by default 1 | Float |
| Design | dashboardAngle | The angle between the dashboard and the plane that is vertical to the user's sight line, by default 0 | Float |
| Design | visibilityUI—dashboard | Whether the dashboard UI is visible, by default true | Boolean |
| Design | visibilityUI—target | Whether the target UI is visible, by default true | Boolean |
| Design | visibilityUI—instruction | Whether the instruction UI is visible, by default true | Boolean |
| Design | Instruction | Instruction text about using the XR application | String |
| Design | safetyZoneDisplayStyle | The hologram style of the safety zone indicator, either "fill" or "outline," by default "outline" | String |
| Control | markerLocationBridge | The location of the registration marker in the crane's bridge dimension (cm) | Float |
| Control | markerLocationTrolley | The location of the registration marker in crane's trolley dimension (cm) | Float |
| Control | markerLocationHoist | The location of the registration marker in crane's hoist dimension (cm) | Float |
| Control | safetyZoneHoist | The range of the safety zone in crane's hoist dimension (cm) | Array of two floats |
| Control | safetyZoneTrolley | The range of the safety zone in crane's trolley dimension (cm) | Array of two floats |
| Control | safetyZoneBridge | The range of the safety zone in crane's bridge dimension (cm) | Array of two floats |
| Control | targetLocationHoist | The location of the target in crane's hoist dimension (cm) | Float |
| Control | targetLocationTrolley | The location of the target in crane's trolley dimension (cm) | Float |
| Control | targetLocationBridge | The location of the target in crane's bridge dimension (cm) | Float |

as the "Ilmatar" crane but located in a different operating environment, thus with a different safety zone, target location, etc. The demo crane DT document corresponds to an *instance* that is adapted from the "Ilmatar" DT document. Similarly, we create a robot arm DT document based on the identified XR ontology in Table 2.

## 3.3 TwinXR-compatible MR applications

The work implemented two TwinXR-compatible MR applications, "HoloCrane" and "HoloRobot." Figures 5, 6 show the two applications in the Unity developer view. The applications are deployed to and running on the Trimble XR10 with HoloLens 2 device in this implementation, but it can also be possibly deployed to other devices such as Meta Quest 2.

The "HoloCrane" application enables users to monitor and control a virtual crane, while the "HoloRobot" application enables users to view the arm robot status, and can be connected to and controlled with external software such as Robot Operating System (ROS). Both applications enable users to view the real-time status of the machine *via* a virtual dashboard, including the real-time data of the crane and target locations along three dimensions, or the orientation and speed of each joint of the arm robot. The linked DT document content is also presented on a dashboard, with basic information about the crane or the robot arm, including its name, description, manufacturer, and owner. In addition, the robot arm application also reads and displays the orientation range, speed range, and offset of each joint from the DT documents. When the *"Move Robot Arm to Origin"* virtual button is triggered, all six joints will be recovered to the offset orientations defined in the DT documents. The defined orientation range and speed range set limits when the robot arm is controlled by external software. With the crane application, users can interact with virtual joysticks for moving the trolley, bridge, and hoist of the crane, or moving the virtual target hologram for target positioning of the crane hook. Both applications share several common functionalities: The safety zone indicator outlines the operationally safe range for the crane movement in three dimensions and therefore improves users' situational awareness. Instructions are available to guide users through the basics of using the application. Interface adjustment allows configuring the visibility of certain holograms. Through scanning the QR code of a Twin ID, the corresponding DT document will be linked to the application.

As shown in Figure 4, the crane application consists of a DT-doc module, two common XR modules, i.e., a control module and a visualization module, as well as the QR code scanning functionality. The robot arm application shares a similar modular architecture.

**TABLE 2** *Descriptions of terms* **for XR ontology in the robot arm DT documents.**

| Category | Term | Description | Format |
|---|---|---|---|
| Design | dashboardPosition | The position of the dashboard center with regard to the user (m), by default (1, 0, 0) | Array of three floats |
| Design | dashboardScale | The scale of the dashboard, by default 1 | Float |
| Design | dashboardAngle | The angle between the dashboard and the plane that is vertical to the user's sight line, by default 0 | Float |
| Design | visibilityUI—dashboard | Whether the dashboard UI is visible, by default true | Boolean |
| Design | visibilityUI—instruction | Whether the instruction UI is visible, by default true | Boolean |
| Design | Instruction | Instruction text about using the XR application | String |
| Design | safetyZoneDisplayStyle | The hologram style of the safety zone indicator, either "fill" or "outline," by default "outline" | String |
| Control | markerLocationX | The location of the registration marker in the pre-defined X dimension (cm) | Float |
| Control | markerLocationY | The location of the registration marker in the pre-defined Y dimension (cm) | Float |
| Control | markerLocationZ | The location of the registration marker in the pre-defined Z dimension (cm) | Float |
| Control | safetyZoneX | The range of the safety zone in the pre-defined X dimension (cm) | Array of two floats |
| Control | safetyZoneY | The range of the safety zone in the pre-defined Y dimension (cm) | Array of two floats |
| Control | safetyZoneZ | The range of the safety zone in the pre-defined Z dimension (cm) | Array of two floats |
| Control | offsetOrientationJoint | The offset orientation of a joint (degree) | Float |
| Control | speedRangeJoint | The speed range of a joint (degree per second) | Array of two floats |
| Control | orientationRangeJoint | The orientation range of a joint (degree) | Array of two floats |

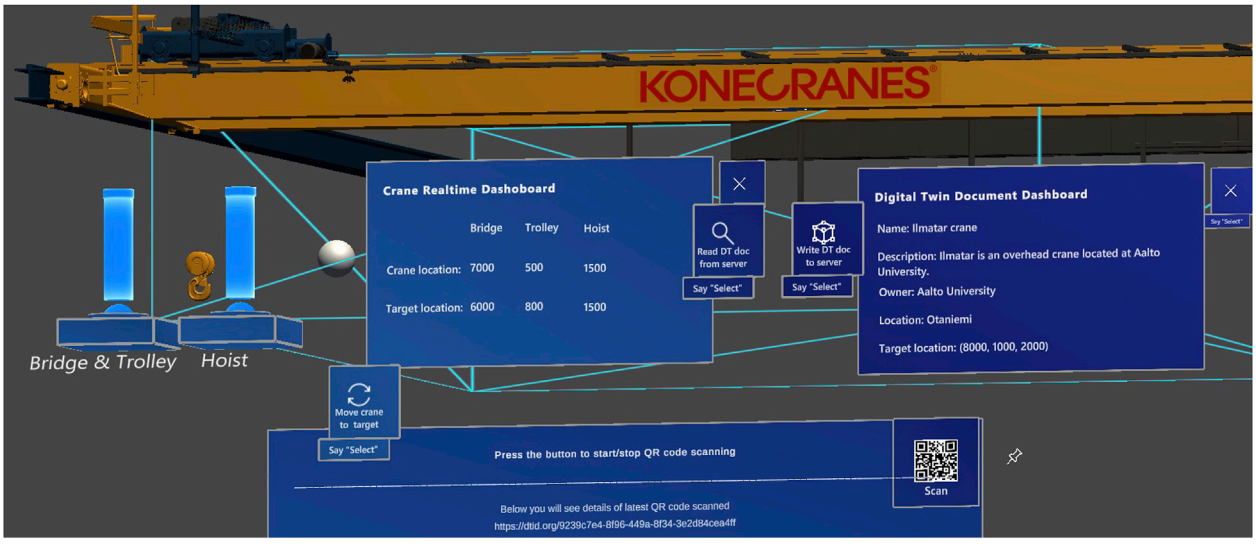


FIGURE 5
Screenshot of the implemented MR application "HoloCrane" (Unity developer view on desktop).

- The DT-doc module includes a DT-doc handler that enables the data flow and conversion of the DT documents on the server and its local copy in the MR application;
- The control module includes the features of virtual joystick control and target positioning. The value of the target position can flow in both directions between the control module and the local DT document;
- The visualization module includes the features of the crane real-time dashboard, the DT document dashboard, the safety zone indicator, the instruction, and the interface adjustment option. The DT document content for the dashboard, the safety zone range for the indicator, the instruction text, and XR component visibility for interface adjustment are determined by the values stored in the local DT document.

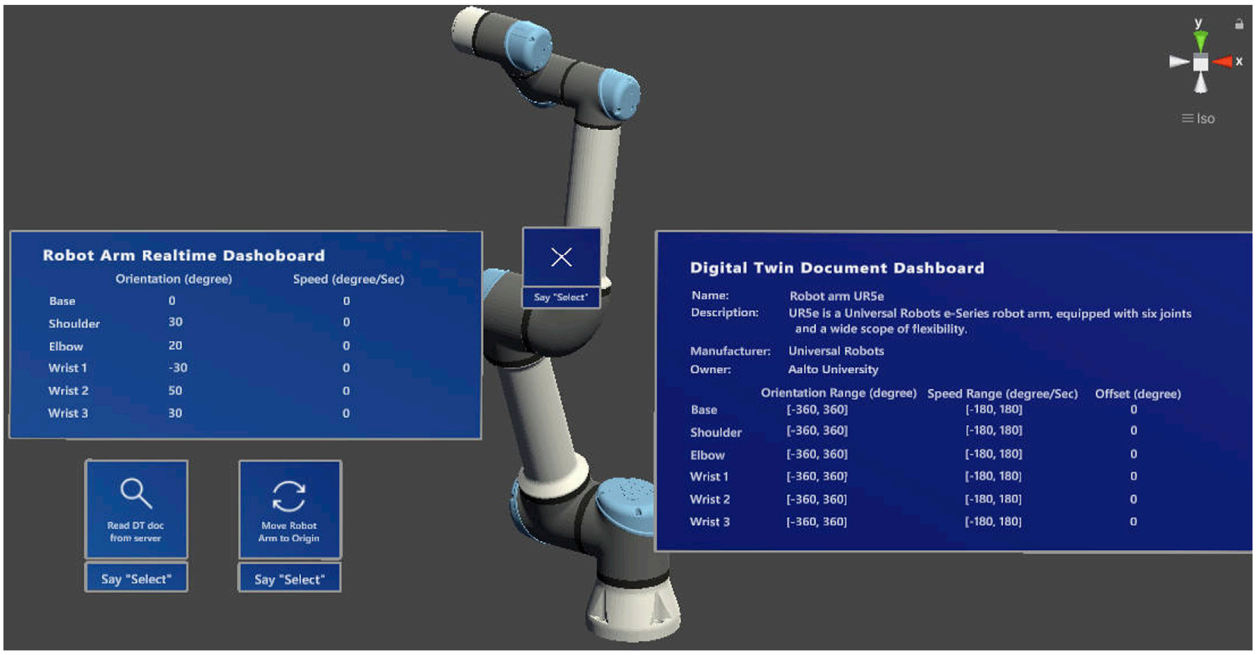


FIGURE 6
Screenshot of the implemented MR application "HoloRobot" (Unity developer view on desktop).

- The QR code scanning functionality enables Twin ID input through scanning the corresponding QR code. The inputted Twin ID is further forwarded to the DT-doc module and used to map the application to the corresponding DT document on the Twinbase server.

# 4 Discussion

This chapter first discusses the advantages of the TwinXR method that advances the synergies between DTs and XR for higher efficiency, scalability, interchangeability, and interoperability. In particular, the advantages of applying the method in the crane and robot arm cases are also discussed to make the justification more concrete. Next, we discuss TwinXR method from the viewpoint of generality. Finally, we address the limitations of the work and propose the future research directions of introducing more detailed principles of the Semantic Web and Knowledge Graphs into the TwinXR method, as well as developing factory-level TwinXR-compatible applications.

## 4.1 Synergies between DTs and XR

The TwinXR method responds to the challenge of synergies between DTs and XR. Through the method, DTs and XR function as enablers for each other to achieve higher efficiency, scalability, interchangeability, and interoperability, which are fundamental characteristics of Industry 5.0 towards the Metaverse. The advantages of the TwinXR method are manifested in both directions: DTs optimize and scale XR application development, while TwinXR-compatible applications also unlock and demonstrate the potential of DTs for data interchange and system interoperation.

### 4.1.1 Efficient and scalable XR application development

This section discusses the efficiency and scalability aspects of using the TwinXR method. The use of DT documents can optimize and scale XR application development by enabling standard central development of an application *origin*, as well as flexible local modification on its *instances*. In other terms, XR solutions can be initiated once, then scaled and delivered widely. The central development of DT document and XR application *origins* requires technical skills in XR development tools like Unity, and domain knowledge on certain machines and their operating environment, while the on-demand composition of XR applications can be conducted solely based on domain knowledge through modifying the established DT document, without knowledge or effort required for XR development.

In addition, this work provides a publicly available Unity package (Tu, 2022b) for data flow and conversion between XR applications and DT documents on the Twinbase server, which eases the application development and use of the TwinXR method. Meanwhile, the composable architecture of a TwinXR-compatible application ensures the autonomy and robustness of the workflows, as each XR component can be modified without affecting the overall structure of the application. Furthermore, the Git-based approach for hosting DT documents ensures the discoverability and reusability of the solutions, as new developers and users can easily harness the repositories of established ones.

The existing XR solutions are rather tailored for a single purpose, while TwinXR-compatible applications can be adapted to new versions according to the requirements of new machines or operating environments that are defined in the corresponding DT documents. Hence, our approach makes it possible to build generic XR solutions for industrial purposes and different applications, as well as

opens up opportunities for multiple types of users from design engineers to machine operators.

### 4.1.2 Data interchange and system interoperation with DTs

This section discusses the interchangeability and interoperability aspects of using the TwinXR method. TwinXR-compatible applications unlock and demonstrate the potential of DTs as an intermediate layer for data integration and information sharing among machines and their linked applications, which enables data interchange and system interoperation.

The shared ontology in DT documents ensures knowledge flow across the domains of meaning and presentation. Through common data ontology, certain fields of a DT document can be shared among several applications linked to one machine instance. Consequently, a TwinXR-compatible application can communicate and work with its linked machine and other applications through the DT documents.

Information-management-oriented DTs can bridge data silos, increase data visibility, and transparency of production and operation processes, as well as boost collaboration across different applications, which ultimately leads to improved efficiency and decision-making in a Smart Factory. The TwinXR method allows bi-directional data flow and conversion between applications and DT documents of machine instances. Viewing and editing a DT document *via* its linked XR application enables intuitive and user-friendly workflows of information presentation and modification.

### 4.1.3 Advantages of using TwinXR in the crane and robot arm cases

This section discusses the advantages of using the TwinXR method in the crane and robot arm cases. We elaborate and compare the two scenarios of developing XR applications with traditional approaches and applying the TwinXR method.

The crane and robot arm in the selected cases represent two common yet diverse machines in a Smart Factory setup. There can typically be multiple cranes or robot arms of the same or similar type in one factory. As an example, our selected case includes the "Ilmatar" crane, and the demo crane of the same type as the "Ilmatar" crane but located in a different operating environment, thus with a different safety zone, and target location. With traditional approaches, one XR application is tailored to one specific machine instance. Developing XR applications for a new crane/robot arm instance requires repeating the process every time. Furthermore, any later updates in the crane/robot arm systems or their operating environment require using specific XR development tools to modify the implementation of the application scripts or scenes. Besides, machine data typically comes in different formats and from different systems in the physical layer. For example, the real-time status of the crane movement can be accessed from its connected OPC UA interface, while the movement data of the robot arm is in ROS. Hence, data mapping between cranes/robot arms and their XR applications is a tedious instance-by-instance process.

Although machines in the physical layer are diverse from each other, they can share common fields of data that are relevant to be viewed, used, or controlled by their XR applications, such as the name, location, movement, and safety zone of the crane or robot arm. The TwinXR method utilizes DT documents to manage and distribute these common data fields, among other heterogeneous machine data, to different applications including the XR interfaces. Following the architecture and workflows of the TwinXR method, a TwinXR-compatible application can be created once and centrally, e.g., for the "Ilmatar" crane, then efficiently scaled to new instances according to new machine or environment specifications, e.g., for the demo crane. This process can be done by simply modifying the associated DT descriptions, without repeating the XR development step. For the crane case, the DT documents of the "Ilmatar" crane and the demo crane share the same data structure and fields with the same or different values, while the DT documents of the crane and robot arm contain different XR ontology. Nevertheless, we can organize and host these DT documents in the same Twinbase server, and follow the same practices of developing and using the TwinXR-compatible applications. This demonstrates TwinXR's capability of enabling generic XR solutions for industrial purposes. In addition, on-demand modification of TwinXR-compatible applications responding to, e.g., the crane/robot arm system updates, or moved location, can be easily done by updating corresponding fields in the DT documents. This process does not require XR development experience, and therefore enables multiple types of users, such as field managers, design engineers, crane/robot arm operators, to directly implement the changes quickly and flexibly.

## 4.2 Generality of the TwinXR method

Overall, TwinXR is designed to be a generic method that leverages DT descriptions in industrial XR applications. To demonstrate its usage and validity, we implement the method for two selected cases with specific setups. Nevertheless, the specifications of the implementation should not affect the overall generality of the method. In other words, we would present TwinXR as a general method that can be adapted to various industrial devices and processes, rather than case studies. The focus of the work is to propose the initial concept and design of the novel method. On the other hand, we do not intend to prove its generality through exhaustive use case implementation. The following discussion will provide empirical evidence for the generality of the method.

First of all, we claim that the TwinXR architecture (Figure 1) is generic. The framework consists of three layers of Smart Factory, DT document, and XR application, with a wide range of options for hardware and software setups proposed. The following content will discuss the generality of each layer.

The Smart Factory layer is designed for a common Smart Factory setup, which contains varied physical machine instances that are connected with DTs and equipped with XR interfaces. In the selected use cases, we involve an industrial crane and a robot arm as reference models, since they are among the most common machines in a Smart Factory. The workflows of the implementation are expected to be easily transferable to other similar cases.

The DT document layer contains documents that describe DTs with common data ontology, thus enabling information management-oriented DTs. For development efficiency, we leverage the specification of DT documents that the authors previously drafted with a Git-based open-source server solution, Twinbase. Yet, the TwinXR method is compatible with different specifications of DT documents. Regardless of the diversity of associated machines, DT documents are designed to contain common fields, such as the basic information of a machine, including its name, description, location, and manufacturer. The generality of DT documents' content is also reflected in the XR parameters, which are categorized into design and

control parameters. Design parameters determine the UI/UX of an XR application, such as the color, shape, size, opacity, and style of XR components, as well as instruction text about using the application. The values of the design parameters should be tailored according to the specifications of machines, operating environments, and processes to meet user-friendliness and safety requirements. Control parameters affect machine operation, such as the marker location, target location, safety zone, moving range, speed range, and offset location of moving components. The control parameters can be shared with and used by other applications that are connected to the same machine. For instance, the cognitive firewall system for Internet of Things proposed in Siegel and Sarma (2019) or the factory control application proposed in Mattila et al. (2022) can also utilize the safety zone parameter from DT documents.

The XR application layer presents TwinXR-compatible XR applications that consist of the DT-doc module and common XR components. We would claim the generality of the composable and modular architecture of the TwinXR-compatible applications, which is independent of specific tools or platforms for XR development and deployment. The DT-doc module enables bi-directional data flow and format conversion between a DT document on the server and its local copy in the XR application. Other modules can read data from the DT document to customize certain XR features. In our implementation, we use the game engine Unity as the development platform and deploy the application to the MR headset Trimble XR10 with HoloLens 2 edition. In addition, we provide a publicly available package to ease the development of TwinXR-compatible applications in the Unity environment. Among various types of user interfaces connected with Smart Factory devices, this work focuses on XR interfaces, including VR, AR, and MR. The specifications that are proposed in the DT document content and the implementation in two industrial use cases are therefore only consider XR. It is nevertheless possible to replace XR with other types of interfaces such as web interfaces in the proposed architecture and workflows. On the other hand, given the future work of introducing the Semantic Web property (which will be elaborated in the next section), it is expected to develop a comprehensive XR ontology for the Smart Factory context. Hence, narrowing down the potential scope of "TwinUI" to "TwinXR" makes ontology development more feasible.

Based on the proposed architecture, we present the workflows of the TwinXR-compatible application development (Figure 3), from creating an origin to adapting it into new instances, with a focus on the transition and linkages between the two workflows. Last but not least, we define the step-by-step processes for the creation and instantiation of a TwinXR-compatible application by developers, as well as the initiation of the application by users. The detailed technical workflows while using an established application are elaborated for two scenarios: one with a TwinXR-compatible application reading a DT document, and the other with an application modifying a DT document. The aforementioned processes and workflows are designed to be applicable in most cases regardless of different setup choices.

We have identified cases in which applying the TwinXR method seems to be unreasonable. For instance, when a type of machine is only used in a specific and static process and location, it might be sufficient to develop its XR application with traditional approaches instead of the flexible yet more costly TwinXR method, as there is no need to further adapt and customize the XR application after the one-time composition. In other words, we need to always consider the aspect of cost-efficiency when applying the TwinXR method, as it requires extra steps of designing the ontology, establishing the DT document layer, and developing a DT-doc module in the XR application.

## 4.3 Future work

The TwinXR method greatly improves interoperability across machines, DT documents, and XR applications. However, the mapping of features between XR components and DT document fields solely relies on the shared data structure of the DT document. This results in the limitation that an XR application *origin* for a certain machine can only be adapted to new *instances* for machines of the same type. To boost the interoperability to the next level, we propose the following directions for future work: introducing the detailed principles and technologies of the Semantic Web and Knowledge Graph, as well as developing factory-level TwinXR-compatible applications.

First, the TwinXR method could be made compatible with the Semantic Web. This work already includes *descriptions of terms* that define the fields of a DT document and provide the context for developers and users to communicate and utilize the ontology, which, however, is readable only by humans. The Semantic Web is a mesh of linked data that can be easily processed by machines (Antoniou and Van Harmelen, 2004), and leveraging its technologies could extend the widescale machine-readability of DT documents. In practice, Twinbase currently uses the YAML format for DT documents because of its user-friendliness. However, YAML is not standardized as a linked data format. To introduce the Semantic Web property, we recommend adopting a standardized linked data format like JSON-LD (Sporny et al., 2020) as the master format of DT documents in future implementations. (It may be possible to use YAML in the future as YAML-LD is currently being specified by the JSON-LD Community Group of W3C (Kellogg et al., 2022).) By making the TwinXR method compatible with the Semantic Web, we will be more ready to move towards the standardization of the XR ontology for DT documents. Besides, future work could introduce the property of Knowledge Graph, the knowledge base that use graph-structured data models or typologies (Ehrlinger and Wöß, 2016). The adoption of Knowledge Graph will facilitate access to and integration of data sources across different machines. Consequently, the composition of TwinXR-compatible applications will become more efficient with minimum manual adaption for new machines and operating environments.

Moving forward with achieving interoperation at an even larger scale across multiple machines, future work could explore the potential of developing factory-level TwinXR-compatible applications. One factory XR application will be linked to one factory DT, which can have multiple machine DTs as its children. In this case, the balance among different DT documents should be investigated. For instance, we shall consider defining operational environment-related fields like safety zone only once in the factory DT space so that they can be accessed by each of its associated machine DTs. At the factory scale, an XR interface will likely be one of the many associated external applications. Hence, a comprehensive semantic description that covers different applications is needed for seamless information exchange (Zillner et al., 2016). With the emerging concept of Smart Factory Web (SFW) (Jung et al., 2017; Heymann et al., 2018), a testbed of the Industrial Internet Consortium for improving factory-to-factory interoperability, we will need to integrate the factory-level

TwinXR method into the cross-site application scenarios with heterogeneous manufacturing infrastructures. The future development of the TwinXR method should consider utilizing existing reference architectures driven by SFW, such as the one proposed in Usländer et al. (2021) for open marketplaces for industrial production.

## 5 Conclusion

This work proposes a novel method TwinXR that uses descriptions of DTs of Smart Factory devices in creating and instantiating industrial XR applications. The motivation comes from the context of human-centric Industry 5.0 towards the Metaverse with immersive digital landscapes to enrich the human experience on factory floors. Interoperability and interchangeability across different components and processes are critical properties of the industrial Metaverse. DTs and XR, two core technological enablers for engineering in the Metaverse, are therefore expected to seamlessly interact and cooperate with each other. Numerous works have indicated the significance of combining DTs and XR in industrial use cases. However, a systematic method is still missing to achieve the synergies between DTs and XR. The TwinXR method addresses this research gap with common data ontology shared among knowledge-based XR interfaces and information-oriented DTs.

We refer to the general concept of a document describing a DT with common data ontology as a DT document. According to the standard (Ala-Laurinaho et al., 2020) used in this work, a DT document contains mandatory fields, such as name, identifier, and description of the DT, as well as optional fields, such as manufacturer, location of the physical product, and connected services. In this work, we focus on the DTs with XR applications as their connected services. Hence, the fields of a DT document are extended to include identified customizable XR features. These XR features cover design parameters that are related to the UI/UX of an XR application, and control parameters that are critical to machine operation. The data that is passed from DTs to XR applications includes both customizable XR features and selected general DT information. The former are used to customize XR applications to fulfill both design and control specifications, while the latter includes Twin ID which is used for the mapping between external applications and DT documents, and other general DT information which is visualized and displayed in XR applications. The data flow and conversion between XR applications and DT documents on the server are proposed to be handled by a DT-doc module in XR applications. The work develops a publicly available Unity package to ease the development of the DT-doc module. We further demonstrate the usage and indicate the validity of the TwinXR method with two industrial use cases in typical Smart Factory setups, involving industrial crane operation, and robot arm control.

The TwinXR method is regarded as a promising way to advance synergies between DTs and XR, as well as a useful tool for both of them: For XR, TwinXR optimizes and scales the XR application development process; For DTs, TwinXR unlocks and demonstrates the potential of DTs for data interchange and system interoperation. We claim that TwinXR is designed to be a generic method that can be adapted to various industrial devices and processes. Future works include introducing more detailed principles and technologies of the Semantic Web and Knowledge Graph, as well as developing factory-level TwinXR-compatible applications.

## Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

## Author contributions

Conceptualization, XT, JA, RA-L, and KT; Methodology, XT, JA, and RA-L; Software, XT and CY; Resources, XT, JA, PS, and KT; Writing—original draft preparation, XT; Writing—review and editing, XT, JA, RA-L, CY, PS, and KT; Visualization, XT, JA, and CY; Supervision, KT; Project administration, PS and KT; Funding acquisition, KT. All authors have read and agreed to the published version of the manuscript.

## Funding

This research was funded by the Business Finland under Grant 3508/31/2019 and ITEA 3 Call 5 MACHINAIDE.

## Acknowledgments

The authors would like to thank all “MACHINAIDE” consortium members and those who presented or participated in discussions of this work.

## Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## Publisher’s note

# References


Ala-Laurinaho, R., Autiosalo, J., Nikander, A., Mattila, J., and Tammi, K. (2020). Data link for the creation of digital twins. *IEEE Access* 8, 228675–228684. doi:10.1109/access.2020.3045856

Antoniou, G., and Van Harmelen, F. (2004). *A semantic web primer*. Cambridge, Massachusetts: MIT press.

Autiosalo, J., Siegel, J., and Tammi, K. (2021). Twinbase: Open-source server software for the digital twin web. *IEEE Access* 9, 140779–140798. doi:10.1109/access.2021.3119487

Autiosalo, J., Vepsäläinen, J., Viitala, R., and Tammi, K. (2020). A feature-based framework for structuring industrial digital twins. *IEEE Access* 8, 1193–1208. doi:10.1109/ACCESS.2019.2950507

Begout, P., Kubicki, S., Bricard, E., and Duval, T. (2022). Augmented reality authoring of digital twins: Design, implementation and evaluation in an industry 4.0 context. *Front. Virtual Real.* 83, 6–11. doi:10.3389/frvir.2022.918685

Breque, M., De Nul, L., and Petridis, A. (2021). *Industry 5.0: Towards a sustainable, human-centric and resilient european industry*. Luxembourg, LU: European Commission, Directorate-General for Research and Innovation.

Brewster, C., and O'Hara, K. (2004). Knowledge representation with ontologies: The present and future. *IEEE Intell. Syst.* 19, 72–81. doi:10.1109/mis.2004.1265889

Chuah, S. H.-W. (2018). Why and who will adopt extended reality technology? Literature review, synthesis, and future research agenda. *Literature Rev. Synthesis, Future Res. Agenda* 2018. doi:10.2139/ssrn.3300469

Ehrlinger, L., and Wöß, W. (2016). Towards a definition of knowledge graphs. *Semant. Posters, Demos, Success.* 48, 2. doi:10.1145/3227609.3227689

Flotyński, J. (2020). *Knowledge-based explorable extended reality environments*. Springer.

Flotyński, J. (2022). Visual aspect-oriented modeling of explorable extended reality environments. *Virtual Real.* 26, 939–961. doi:10.1007/s10055-021-00601-7

Göbel, M. (2022). SimpleJSON. Available at: https://github.com/Bunny83/SimpleJSON (Accessed July 26, 2022).

Górski, F., Buń, P., Zawadzki, P., and Wichniarek, R. (2019). "Knowledge management in open industrial virtual reality applications," in *International scientific-technical conference MANUFACTURING* (Springer), 104–118.

GS1 (2021). GS1 web vocabulary. Available at: https://www.gs1.org/voc/ (Accessed July 25, 2022).

Heymann, S., Stojanovci, L., Watson, K., Nam, S., Song, B., Gschossmann, H., et al. (2018). Cloud-based plug and work architecture of the iic testbed smart factory web. IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA) (IEEE) 04-07 September 2018, Turin, Italy, IEEE, 1, 187–194.

Jacoby, M., and Usländer, T. (2020). Digital twin and internet of things–current standards landscape. *Appl. Sci.* 10, 6519. doi:10.3390/app10186519

Jung, J., Song, B., Watson, K., and Usländer, T. (2017). "Design of smart factory web services based on the industrial internet of things," in *Proceedings of the 50th Hawaii international conference on system sciences*. doi:10.24251/HICSS.2017.716

Kaebisch, T., KamiyaMcCool, M., Charpenay, V., and Kovatsch, M. (2020). Web of things (WoT) thing description. Available at: https://www.w3.org/TR/wot-thing-description. (Accessed July 25, 2022).

Kellogg, G., Polli, R., Scherbakov, A., and Thibodeau, T. (2022). YAML-LD. Available at: https://github.com/json-ld/yaml-ld (Accessed August 04, 2022).

Lee, L.-H., Braud, T., Zhou, P., Wang, L., Xu, D., Lin, Z., et al. (2021). *All one needs to know about metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda. arXiv preprint arXiv:2110.05352*

Liu, M., Fang, S., Dong, H., and Xu, C. (2021). Review of digital twin about concepts, technologies, and industrial applications. *J. Manuf. Syst.* 58, 346–361. doi:10.1016/j.jmsy.2020.06.017

Longo, F., Padovano, A., and Umbrello, S. (2020). Value-oriented and ethical technology engineering in industry 5.0: A human-centric perspective for the design of the factory of the future. *Appl. Sci.* 10, 4182. doi:10.3390/app10124182

Ma, X., Tao, F., Zhang, M., Wang, T., and Zuo, Y. (2019). Digital twin enhanced human-machine interaction in product lifecycle. *Procedia Cirp* 83, 789–793. doi:10.1016/j.procir.2019.04.330

Mattila, J., Ala-Laurinaho, R., Autiosalo, J., Salminen, P., and Tammi, K. (2022). Using digital twin documents to control a smart factory: Simulation approach with ros, gazebo, and twinbase. *Machines* 10, 225. doi:10.3390/machines10040225

Microsoft (2020). Digital twin Definition Language. Available at: https://github.com/Azure/opendigitaltwins-dtdl (Accessed July 25, 2022).

Plattform Industrie 4.0 (2020). Details of the asset administration Shell. Available at: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_o% f_the_Asset_Administration_Shell_Part1_V3.pdf?_blob=publicationFile&v=5 (Accessed July 25, 2022).

Podder, A., Gruchalla, K., Brunhart-Lupo, N., Pless, S., Sica, M., and Lacchin, P. (2022). Immersive industrialized construction environments for energy efficiency construction workforce. *Front. Virtual Real.* 29. doi:10.3389/frvir.2022.781170

SAREF (2021). Available at: https://saref.etsi.org/ (Accessed July 25, 2022).

Schema.org (2022). Available at: https://schema.org/ (Accessed July 25, 2022).

Siegel, J., and Sarma, S. (2019). A cognitive protection system for the internet of things. *IEEE Secur. Priv.* 17, 40–48. doi:10.1109/msec.2018.2884860

Sparkes, M. (2021). What is a metaverse. *New Sci.* 251, 18. doi:10.1016/S0262-4079(21)01450-0

Sporny, M., Longley, D., Kellogg, G., Lanthaler, M., Champin, P.-A., and Lindström, N. (2020). JSON-LD 1.1. Available at: https://www.w3.org/TR/json-ld/ (Accessed August 4, 2022).

Tu, X., Autiosalo, J., Jadid, A., Tammi, K., and Klinker, G. (2021). A mixed reality interface for a digital twin based crane. *Appl. Sci.* 11, 9480. doi:10.3390/app11209480

Tu, X., and Autiosalo, J. (2022). TwinXR: Twinbase for industrial XR use cases. Available at: https://github.com/Plan-T42/TwinXR-TwinBase-for-Industrial-XR-Use-Casess.

Tu, X. (2022a). TwinXR: HoloCrane unity demo. Available at: https://github.com/Plan-T42/TwinXR-HoloCrane-Unity-Demo (Accessed July 26, 2022).

Tu, X. (2022b). TwinXR: Unity package of DT-doc handler. Available at: https://github.com/Plan-T42/TwinXR-DT-doc-handler-package (Accessed July 26, 2022).

Usländer, T., Schöppenthau, F., Schnebel, B., Heymann, S., Stojanovic, L., Watson, K., et al. (2021). Smart factory web–a blueprint architecture for open marketplaces for industrial production. *Appl. Sci.* 11, 6585. doi:10.3390/app11146585

Vogel-Heuser, B., and Hess, D. (2016). Guest editorial industry 4.0–prerequisites and visions. *IEEE Trans. Automation Sci. Eng.* 13, 411–413. doi:10.1109/tase.2016.2523639

Weistroffer, V., Keith, F., Bisiaux, A., Andriot, C., and Lasnier, A. (2022). Using physics-based digital twins and extended reality for the safety and ergonomics evaluation of cobotic workstations. *Virtual Real.* 3, 781830. doi:10.3389/frvir.2022.781830

Yang, C., Tu, X., Autiosalo, J., Ala-Laurinaho, R., Mattila, J., Salminen, P., et al. (2022). Extended reality application framework for a digital-twin-based smart crane. *Appl. Sci.* 12, 6030. doi:10.3390/app12126030

Zhu, Z., Liu, C., and Xu, X. (2019). Visualisation of the digital twin data in manufacturing by using augmented reality. *Procedia Cirp* 81, 898–903. doi:10.1016/j.procir.2019.03.223

Zillner, S., Ebel, A., and Schneider, M. (2016). Towards intelligent manufacturing, semantic modelling for the steel industry.**The research leading to these results has received funding from the European Community's Research Fund for Coal and Steel (RFCS) under grant agreement n° CT 2012 00038 (I2MSteel). *IFAC-PapersOnLine* 49, 220–225. doi:10.1016/j.ifacol.2016.10.124

Zuehlke, D. (2010). SmartFactory – towards a factory-of-things. *Annu. Rev. control* 34, 129–138. doi:10.1016/j.arcontrol.2010.02.008