Journal of Intelligent Manufacturing (2021) 32:501–544
https://doi.org/10.1007/s10845-020-01586-x

VREDI: virtual representation for a digital twin application in a work-center-level asset administration shell

Kyu Tae Park[1,2] · **Jinho Yang**[1] · **Sang Do Noh**[1]

Received: 17 May 2019 / Accepted: 7 May 2020 / Published online: 17 May 2020

Abstract

The asset administration shell (AAS) has a virtual representation as an asset description and technical functionality as a smart manufacturing service. A digital twin (DT) is an advanced virtual factory technology that has simulation as its core technical functionality, which it performs in the type and instance stages of the physical asset. For providing an efficient information object to the DT application, this paper proposes Virtual REpresentation for a DIgital twin application (VREDI): an asset description for the operation procedures of a work-center-level DT application. For the successful application of DT as a smart factory technology, VREDI is designed to meet four core technical requirements—DT definition, AAS property inheritance, improving the existing asset description, and supporting DT-based technical functionalities. Based on the analysis of the technical requirements, the elements of VREDI are derived and the reference relationships between them are designed. It is then possible to provide the required technical functionality using the VREDI header, and a detailed P4R structure and elements of the body are defined. VREDI is applied to the concept to support the main properties of the DT. It is designed to inherit the AAS properties for efficient information management and interoperability. The application of advanced concepts such as "type and instance" and supporting vertical integration and horizontal coordination overcomes the limitations of the existing asset descriptions. Additionally, VREDI designates elements for supporting six DT-based technical functionalities in the type and instance stages of the physical work center.

Keywords Asset description · Digital twin · Digital-twin-based technical functionality · Service-oriented architecture · Virtual representation · Work-center-level asset administration shell

Abbreviations

AAS	Asset administration shell
API	Application programming interface
BOM	Bill of materials
CMSD	Core manufacturing simulation data
CDL	Configuration data library
CNC	Computerized numerical control
CPPS	Cyber physical production system
CPS	Cyber physical system
CSPI	Commercial off-the-shelf simulation package interoperability
DES	Discrete event simulation
DDL	Data description language
DT	Digital twin
I4.0	Industrie 4.0
ICT	Information and communication technology
ID	Identifier
IIoT	Industrial internet of things
IoT	Internet of things
MFC	Microsoft foundation class
MHC	Material handling conveyor
MHE	Material handling equipment
MHR	Material handling robots
MHV	Material handling vehicle
MMS	Modular manufacturing system
MSF	Micro smart factory
MTBF	Mean time between failures
MTTR	Mean time to repair
NESIS	Neutral simulation schema
P4R	Product, process, plan, plant, and resource
PLC	Programmable logic controller
RAMI	Reference architectural model industrie
REST	Representational state transfer

✉ Sang Do Noh
sdnoh@skku.edu

[1] Department of Industrial Engineering, Sungkyunkwan University, Suwon-si, Republic of Korea

[2] Digital Factory Solution Center, MICUBE Solution Inc., Seoul, Republic of Korea

RBR	Rule-based reasoning
SOA	Service-oriented architecture
SOAP	Simple object access protocol
STEP	Standard for the exchange of product
UML	Unified modeling language
VREDI	Virtual representation for a digital twin application
WCF	Windows communication foundation
WIP	Work in process
XML	Extensible markup language

Introduction

The fourth industrial revolution is driving global industrial innovations; for example, Germany's "I4.0," South Korea's "The Manufacturing Innovation Strategy 3.0," the United States' "Advanced Manufacturing Partnership," and China's "Made-in-China 2025" (Adolphs et al. 2016; Cadavid et al. 2020; Derigent et al. 2020; Jeon et al. 2020; Li 2018; Wiktorsson et al. 2018). These innovation strategies in the manufacturing industry have a common goal of implementing smart factories, in which manufacturing technology is merged with ICT (Cadavid et al. 2020; Oztemel and Gursev 2020; Wiktorsson et al. 2018). Thus far, a smart factory is not a fully established concept, and consequently, it has various definitions. One such definition states that a smart factory is "a fully integrated and collaborative manufacturing system that responds in real time to meet the changing demands and conditions of a factory, supply network, and customer needs" (Thompson 2014). Furthermore, the planning and development of smart factories is progressing in two major research directions: advanced manufacturing operation and personalized production for service refinement (Wiktorsson et al. 2018). Advanced operation of the manufacturing field is expected to be a solution for addressing the performance hurdle in personalized production. Therefore, the provisioning of advanced technical functionalities is key to satisfying the abovementioned two directions (Kim et al. 2020; Park et al. 2020).

RAMI 4.0 is among the various reference models that have been proposed for realizing a smart factory (Adolphs et al. 2016; Cadavid et al. 2020; Weber et al. 2017). This model complies with the characteristics of SOA and is composed of three dimensions for realizing a smart factory: (1) a hierarchy level, (2) value stream, and (3) layers (Adolphs et al. 2016; Bedenbender et al. 2017; Suri et al. 2017). As a core element, RAMI 4.0 has an AAS, which contains the definitions of the information and functionalities of the manufacturing field. The AAS has a virtual representation as an asset description, and technical functionality as a smart manufacturing service (Adolphs et al. 2016). The virtual representation and technical functionalities have been proposed conceptually as definitions for various requirements; however, the development of implementation cases or reference models is lacking (Adolphs et al. 2016). The role of the AAS is indispensable in the implementation of smart factories that comply with RAMI 4.0. Thus, the development of reference models for the virtual representation and establishment of cases relating to the technical functionality are necessary.

A DT has gained increasing research interest as a core technology for advancing operations in smart factories. In the manufacturing field, a DT is a virtual factory with simulation as its core technical functionality (Cheng et al. 2018; Grieves 2014; Liu et al. 2019; Tao et al. 2019). The DT is required to achieve vertical integration with the physical manufacturing asset and horizontal coordination with advanced engineering applications (Alam and El Saddik 2017; Gabor et al. 2016; Park et al. 2019a, b; Tao et al. 2018; Uhlemann et al. 2017; Wiktorsson et al. 2018). The DT represents the configuration of the physical manufacturing asset, reflects the functional units of each element in the asset, synchronizes information object based on vertical integration and horizontal coordination through virtual representation, and provides technical functionalities through prediction, diagnosis, and coordination of the field (Alam and El Saddik 2017; Cheng et al. 2018; Grieves 2014; Redelinghuys et al. 2019; Tong et al. 2019). In addition, the DT technology can be used in the AAS of RAMI 4.0. The existing asset description of a work center—an asset that has a hierarchy level between a station and an enterprise in RAMI 4.0—can be used for rapid and accurate modeling work, and it enables horizontal coordination with engineering applications. However, these descriptions are insufficient for creating and synchronizing a DT that includes advantages such as advanced levels of coordination and vertical integration, aggregation in a distributed environment, and the capability of enhancing the efficiency of the asset lifecycle (Bloomfield et al. 2012; IEC 2016; Lee et al. 2011, Lee et al. 2012; Riddick and Lee 2010; SISO 2010; Mónica et al. 2016).

This paper presents the VREDI description method, which is the proposed asset description of a DT that meets four core technical requirements (DT definition, AAS property inheritance, improving existing asset description, and supporting DT-based technical functionalities) for VREDI to apply the DT to a work-center-level AAS. The requirements of the DT definition ensure that the VREDI is designed such that it is suitable for the DT; the AAS property inheritance reduces information duplication, thereby enabling efficient information management and operation of the VREDI. Further, the VREDI overcomes the limitations of the existing asset description and supports DT-based technical functionalities required for a work-center-level DT. In addition, VREDI is an advanced object-oriented model that has *Product, Process*, *Plan*, *Plant*, and *Resource* as elements, and it applies the "type and instance" concepts. Further, it is

designed to meet the requirements of work-center-level AAS in RAMI 4.0 and is an early case of the application of the concept of AAS. In this study,

I. The four core requirements for appropriately applying a VREDI to DT application in work-center-level AAS were analyzed.
II. Six DT-based technical functionalities for new production methods based on a DT were listed to describe the role of a DT in the work-center-level AAS and to extract the dimension of the technical requirements.
III. An architectural framework of the DT application, which allows the VREDI to provide the input information object for creating and synchronizing the DT, was designed.
IV. The core elements of *Product, Process*, and *Resource* (the type classes of VREDI), and *Plan* and *Plant* (the instance classes of VREDI), and the relationships among the classes were defined.
V. The elements of the classes were discussed in detail and the manifests of these elements were summarized;
VI. The proposed VREDI and DT application were implemented and applied in case studies and its effects were verified.

Research background

RAMI 4.0

Multilayered reference architectures or models with respect to the composition of a system have been proposed for the successful utilization of a smart factory (da Cruz et al. 2018; Redelinghuys et al. 2019; Weber et al. 2017). Fundamentally, the application of a reference architecture is adopted based on the target technology and business and technical requirements (da Cruz et al. 2018; Xu et al. 2014). Among these models, the most popular is RAMI 4.0, which was proposed by BITCOM, VDMA, and ZWEI (Adolphs et al. 2016; Derigent et al. 2020; Dorst 2015; Hankel and Rexroth 2015; Weyrich and Ebert 2015; Zezulka et al. 2016). This reference model is a three-dimensional model that reflects technical and economic properties (Adolphs et al. 2016; Weyrich and Ebert 2015). RAMI 4.0 simplifies the major aspects of different stakeholders of I4.0 and defines the general guidelines for the three axes and functions that need to be performed (Bedenbender et al. 2017; Oztemel and Gursev 2020; Suri et al. 2017). The three axes are the hierarchy levels, value stream, and layers, which have the following characteristics (Adolphs et al. 2016; Fleischmann et al. 2016; Kagermann et al. 2013).

- *Hierarchy Levels* Realize the allocation of the functionality of components to hierarchy levels.
- *Value Stream (Life Cycle)* Allow classification according to the status of the life cycle and distinguish between the life cycles of type and instance.
- *Layers* Separate concerns regarding interoperability and collective understanding of syntax and semantics to different perspectives and play the role of an interface between the physical and cyber worlds.

Furthermore, RAMI 4.0 has an AAS with virtual representation and technical functionality as the core elements, and this AAS has a 'manifest,' which is the metadata of data contents, and a 'component manager,' which supports external approaches, as core components (Adolphs et al. 2016). The manifest and component manager enable the approach of a loosely coupled SOA. The AAS that meets the requirements summarized in Table 1 enables the existing components to function as I4.0-components (Adolphs et al. 2016; Chuang 2016; Hankel and Rexroth 2015). Furthermore, the AAS should be able to define and use all information and functions (Adolphs et al. 2016).

SOA

An SOA is defined as "a design framework for the construction of a computational system by a 'combination of services'" (Komoda 2006; Ramollari et al. 2007). It is difficult to find a unified definition for 'service,' but it can be explained as an abstract element that provides capabilities according to the user's needs (Komoda 2006; MacKenzie et al. 2006; Ramollari et al. 2007). A system design in the SOA includes consideration as a service provider to effectively deliver capabilities to service consumers (MacKenzie et al. 2006; Perrey and Lycett 2003). Furthermore, an SOA is an architectural design framework that is most notable in architectures using the IoT (Xu et al. 2014). Internal and external properties are among the main properties of an SOA. They are defined as follows:

- *Internal property* An SOA is designed and driven based on data created from distributed heterogeneous elements, and it requires considering the internal organization to provide an integrated solution (Komoda 2006; Papazoglou et al. 2007). Further, the SOA enables heterogeneous distributed applications to achieve adaptive, flexible, and extensible development, integration, management, and replacement through 'loosely-coupled' connections (Bandyopadhyay and Sen 2011; Papazoglou et al. 2007; Yaqoob et al. 2017). In this case, they form a centralized coordination element that aggregates the operations of information (Bandyopadhyay and Sen 2011; Komoda

Table 1 Principles for the administration shell and individual properties, data, and functions (Adolphs et al. 2016)

Index	Contents
1	Virtual representation in the administration shell comprises a body and header
2	Body contains information about the respective asset
3	Header contains information about the utilization of the asset
4	Administration shell consists of the manifest and component manager
5	Information in the administration shell is accessible by means of a service-oriented architecture (API)
6	Virtual representation represents information about different application aspects of the asset
7	Structuring according to views
8	Administration shell has a unique ID
9	Asset has a unique ID
10	Work center is also an asset; it has an Administration Shell and is accessible by means of an ID
11	Types and instances must be identified as such
12	Administration shell can include references to other administration shells or RAMI 4.0 information
13	Additional properties, e.g., manufacturer-specific, must be possible
14	A reliable minimum number of properties must be defined for each administration shell
15	The properties and other elements of information in the administration shell must be suitable for the types and instances
16	There must be a capability of hierarchical and countable structuring of the properties
17	Properties must be able to reference other properties and in other administration shells
18	Properties must be able to reference data and the functions of (or at least within) the administration shell

2006). The application of the centralized coordination element, which is also called a 'service bus,' can save the time and cost required for each step of the lifecycle from the development of the SOA to its maintenance, and it can provide more effective services compared to the use of a conventional architecture taxonomy such as point-to-point (Bandyopadhyay and Sen 2011; Komoda 2006; Zhang et al. 2015).

- *External property:* The internal property of the SOA leads to a change in properties for the non-software elements in the architecture. All elements should be allowed to use the Web or network to provide a centralized aggregation of distributed elements and an integrated solution (MacKenzie et al. 2006). The SOA is the most frequently used design framework in the architectural design of IIoT-based applications because it uses the Internet and provides high-level interoperability, heterogeneity, and integration owing to the abovementioned internal properties (da Cruz et al. 2018; Xu et al. 2014; Yaqoob et al. 2017; Yoon et al. 2019; Zhang et al. 2015). Thus, in an SOA, independent applications do not perform all specific roles; however, services are configured and provided on the basis of the service bus, which is a centralized coordination element (da Cruz et al. 2018; MacKenzie et al. 2006; Yoon et al. 2019; Xu et al. 2014; Zhang et al. 2015).

DT and DT-based technical functionality

DT was first introduced as an engineering term by Grieves in 2003, and its concepts have evolved through continued research (Grieves 2014; Liao et al. 2018; Ngai et al. 2008). A DT can be defined as an advanced virtual model that represents the heterogeneous elements and functional units of a physical asset, and it reflects an information object (Grieves 2014; Cheng et al. 2018; Liu et al. 2019). In the manufacturing field, it can be defined as an advanced virtual factory that represents the heterogeneous configuration and functional units of a physical manufacturing asset and synchronizes the information object. Its core technical functionality is a simulation related to the lifecycle of an asset, such as design, operation, and production (Alam and El Saddik 2017; Cheng et al. 2018; Grieves 2014; Park et al. 2019a, b; Tong et al. 2019; Lim et al. 2019; Vachálek et al. 2017; Schleich et al. 2017). A DT is an advanced concept compared to existing virtual simulation models, and it has relatively sophisticated properties (Alam and El Saddik 2017; Gabor et al. 2016; Grieves 2014; Park et al. 2019a, b; Uhlemann et al. 2017). These properties are

- Automatic creation with pre-defined configurations and functional units
- Transmission or reception of information from a site through vertical integration

- Advanced process that applies horizontal coordination to advanced engineering applications
- Repeated derivation of performance indicators for prediction and diagnosis.

Based on the abovementioned properties of the DT, the manufacturing industry can improve the accuracy of management and improve the efficiency of decision-making or asset operation (Qi and Tao 2018; Tong et al. 2019; Tao and Zhang 2017). Further, the DT can achieve cyber-physical integration of the smart manufacturing paradigm by playing the role of the core technology of the CPS for the design and operation stages of the work center (Cheng et al. 2018; Gabor et al. 2016; Qi and Tao 2018; Park et al. 2020). Moreover, as a concept that can respond to the entire product lifecycle, a DT can upgrade the entire manufacturing process and system (Adolphs et al. 2016; Tong et al. 2019; Qi and Tao 2018; Wiktorsson et al. 2018). The case studies of DT-based technical functionalities include manufacturing system design (Tao et al. 2019); device execution, planning, and scheduling (Zhang et al. 2017; Park et al. 2020); resource allocation (Tao and Zhang 2017); predictive maintenance and quality prediction (Wiktorsson et al. 2018); abnormal situation notification (Park et al. 2019b, 2020); time-machine monitoring (Park et al. 2019b); diagnosis; and dynamic and proactive response (Park et al. 2019a, 2020).

Asset description of the virtual factory technology

The core element for the successful application of a DT is an asset description, which is connected to the field. Although studies have been conducted on the asset description of DT, workers, and facilities, studies on the work center are lacking. Furthermore, detailed description of every element in the models was not included in the studies (Nikolakis et al. 2019; Qi et al. 2018; Schroeder et al. 2016). However, conventional asset descriptions for representing virtual work centers have been proposed for work-center-level perspective (Cheng et al. 2018; SISO 2012). These virtual representations are designed for high-level interoperability and accurate field description, and they are implemented through DDLs such as XML and UML (SISO 2012).

Table 2 compares the results of CMSD (Bloomfield et al. 2012; Riddick and Lee 2010; Mónica et al. 2016), CSPI (SISO 2010), NESIS (Lee et al. 2011), and P3R (Lee et al. 2012), which are information models used to describe a simulation or work center at the work-center-level. Studies on CMSD and its upgrade (Bloomfield et al. 2012; Riddick and Lee 2010; Mónica et al. 2016) and P3R (Lee et al. 2012) examined rapid modeling and coordination based on predefined components and functions among the four characteristics of a DT in comparison to the conventional virtual simulation model. However, they did not investigate

Table 2 Comparison of asset descriptions from the work-center perspective

Name	Scope	Coverage	Vertical integration	Horizontal coordination	Remarks
CMSD	Simulation of manufacturing operations in job shop environment	Part information, layout, resource information, production operation, production planning, and support	No specific support for data exchange with real work center	No specific support for data exchange with applications	Major attributes are clearly defined
CSPI	Support interoperability between two or more different simulation packages	Entity transfer, shared resource, shared event, shared data structure	No specific support for data exchange with real work center	Support for data exchange between two or more simulations	Only certain key attributes are named and defined
NESIS	Exchange of simulation models between simulation and other manufacturing applications	Product, process, resource, Sim_List, configuration	No specific support for data exchange with real work center	Support for data exchange between simulation and manufacturing applications	Major attributes are clearly defined. No application of the "type and instance" concept
P3R	Simulation of material flow analysis	Product, process, plant, resource	No specific support for data exchange with real work center	No specific support for data exchange with applications	Major attributes are clearly defined No application of the "type and instance" concept

advanced decision-making or asset control through vertical integration using the status, dynamic prediction, and diagnosis. CSPI and NESIS pursued high horizontal coordination between simulations or between simulation and engineering applications; however, it exhibited limitations in terms of vertical dynamic coordination with the field. Furthermore, they did not achieve optimal information management and utilization using aggregation in the distributed environment of RAMI 4.0 (Lee et al. 2011; SISO 2010).

VREDI for a DT application in work-center-level AAS

Four core technical requirements for VREDI, and DT-based technical functionality at work-center-level

VREDI is designed to meet four core technical requirements of the appropriate asset description for DT. The four dimensions of the technical requirement are shown in Fig. 1. VREDI should satisfy the definition of DT in a work-center-level AAS. As described in Sect. 2.3 and shown in Fig. 1a, VREDI must meet the following requirements.

I. Provide efficient information for creating a DT automatically with a CDL. With the components of the DT application, the object of VREDI is implemented as input information for creating, synchronizing, and utilizing a DT of the work-center-level in RAMI 4.0.
II. Vertical integration represents the characteristic that it can be operated and integrated based on one virtual representation from the asset layer at the bottom to the enterprise layer at the top. Therefore, VREDI should contain current site information objects of the physical asset for supporting the vertical integration feature of the DT.
III. Horizontal coordination represents the coordination between the DT and various engineering applications. The VREDI should provide information objects

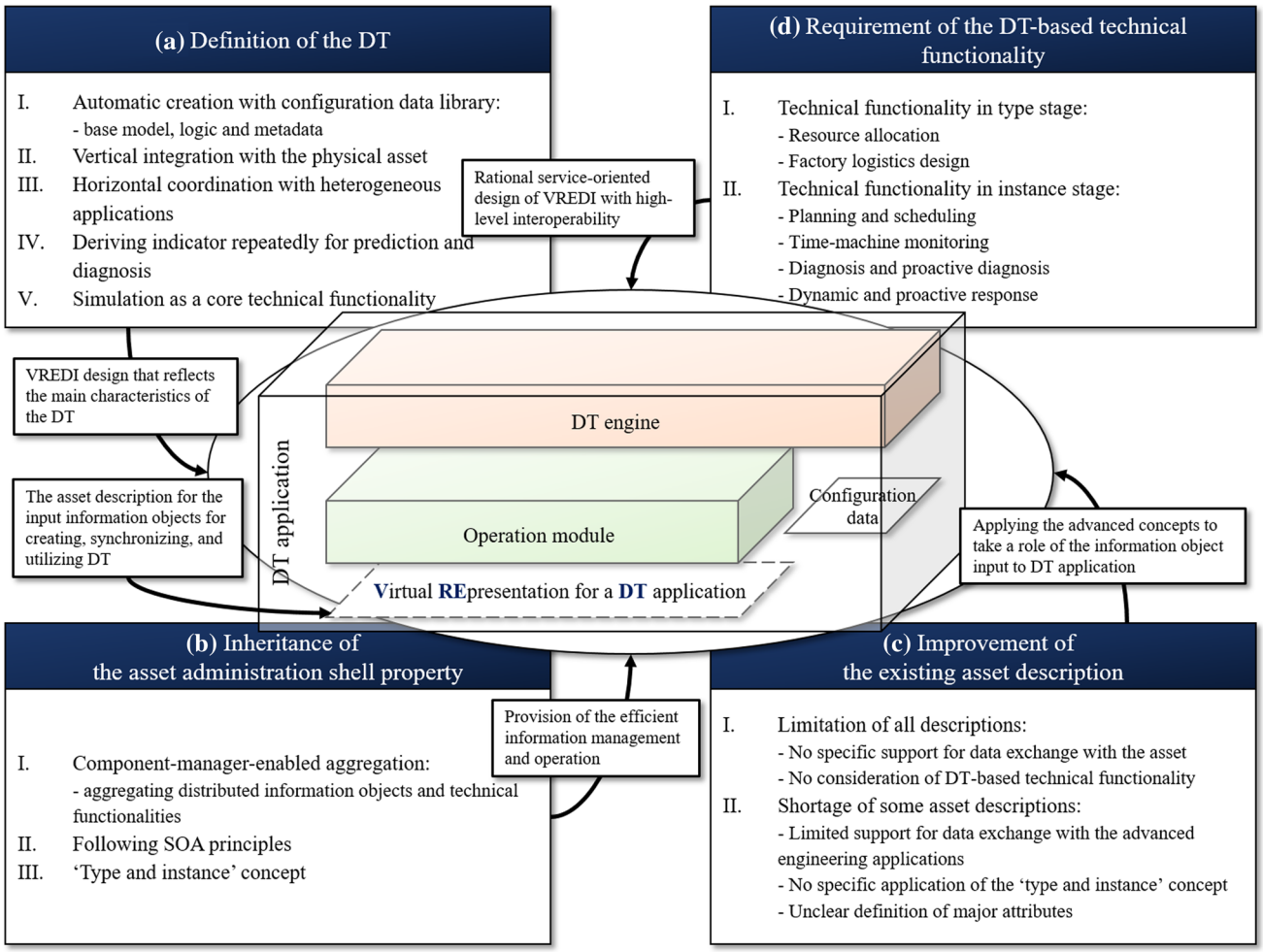

Fig. 1 Conceptual map of VREDI

that can reference the technical functionalities from other advanced engineering applications for supporting horizontal coordination.

IV. The DT should derive performance indicators repeatedly with simulation, which is the core technical functionality of the DT. Therefore, the VREDI needs to support various types of simulation where the component manager is contracted.

VREDI inherits the property of the work-center-level AAS for efficient information management and high-level interoperability, as shown in Fig. 1b and Table 1. VREDI must be able to meet the following requirements. The work center is located between the station-level and the enterprise-level, and it is an asset that processes and assembles the delivered parts and raw materials, which are delivered by over-the-counter delivery. Further, VREDI delivers over-the-counter products that meet the specifications required by customers or other entities in the value chain. In addition, VREDI needs to be designed to satisfy all AAS properties:

I. Enable distributed information management and operation according to component-manager-enabled aggregation. Component-manager-enabled aggregation represents the characteristic of aggregating the required information objects and technical functionalities in a variable manner by accessing other AAS at the work center or other levels such as at the product, field device, control device, station, and enterprise. The component-manager-enabled aggregation can achieve information exchange through communication between layers and applications; the AAS is the information exchange between VREDI rather than direct communication between the DT that has been constructed on the basis of the VREDI.

II. VREDI can achieve loosely-coupled integration using the component manager and manifest those by following the SOA principles. Because loosely-coupled integration does not require tightly-coupled integration, which is not suitable in a heterogeneous development environment, the loosely-coupled integration can improve application and integration efficiency. When using a web service, the service host simply receives information and returns the technical functionality of the engineering application when the component manager contacts the engineering application for service composition. For enabling efficient loosely-coupled integration, the VREDI should be designed to consider this service composition. Therefore, the VREDI needs elements that represent the technical functionality of other advanced engineering applications.

III. The "type and instance" concept should be applied to decrease duplicate information usage. Type information describes static element type information, which refers to static information that does not change according to the situation in the work center operation, and instance information refers to information that may vary depending on the situation. In this case, type information is referenced as instance information, and therefore, it is possible to describe both unchanged information and information on various elements in the work center. Only changed information can be stored for each element in the work center and static information can be used in the reference; therefore, it is possible to prevent duplication.

In order to advance from a conventional virtual factory to DT, VREDI is required to improve the existing asset descriptions as shown in Fig. 1c and Table 2, and detailed as follows:

I. The existing asset descriptions do not contain the element to support vertical integration and DT-based technical functionality. VREDI should include elements around the current asset operation for meeting the advancement from virtual factory to DT. In addition, VREDI should contain the information object that enables the DT-based technical functionality.

II. VREDI should support the horizontal coordination with advanced engineering applications so that the advanced asset operation is enabled. To reduce usage of duplicated information, in VREDI, the "type and instance" concept should be applied with clear definitions of major attributes.

There are six DT-based technical functionalities in which work-center-level DT is utilized, as shown in Fig. 1d. Two technical functionalities exist in the work center design stage, which is the type stage, and four technical functionalities exist in the work center operation stage, which is the instance stage. The most appropriate technical functionalities provided by the work-center-level AAS were selected from among the cases of DT-based technical functionalities surveyed in Sect. 2.3. These are solutions for operational advances or decision-making support for the discrete operation aspect of the physical work center. The technical functionalities for all levels of AAS can be included in the AAS of the corresponding hierarchy levels; the component manager in the work-center-level AAS with the DT application accesses these information objects and technical functionalities. Hence, the technical functionalities of the AAS at different levels can be coordinated. The technical functionalities of the type stage in the work-center-level aspect and its requirements for VREDI are as follows:

I. 'Resource allocation': This technical functionality determines the number of various manufacturing resources. When this technical functionality is considered, the VREDI should provide information on the allocation plan of the configuration of the physical asset to the DT application.
II. 'Factory logistics design': This technical functionality designs the work center logistics. It allows VREDI to include information objects on the systemic configuration of the physical asset to represent the logistics plan of the configuration of the physical asset in the DT.

The operation stage of the work center can be divided further into type and instance phases, which are the operation planning and operation execution phases, respectively. The technical functionalities of the operation stage in the work-center-level aspect and its requirements for VREDI are given below.

III. 'Planning and scheduling': This technical functionality derives the production plan or dispatching sequence by reflecting the status of the work center. This technical functionality—used in the operation planning phase—must be connected to an application that derives the production plan and sequence by date within the capacity of the field. Therefore, VREDI is designed to aggregate the production plan and sequence plan alternatives for deriving an appropriate performance indicator.
IV. 'Time-machine monitoring': This technical functionality traces the history, monitors the present operation, predicts the future of the operation of the physical asset, and delivers the information to the user. The DT synchronizes past production history information to reproduce asset operation for tracing history, and the current operation and situation information of the physical asset to monitor the present. In addition, the DT executes the simulation based on the current operation information to show an asset operation in the future. This technical functionality—used in the operation execution phase—allows the VREDI object to re-aggregate, and makes predictions in accordance with the time and asset desired by the user.
V. 'Diagnosis and proactive diagnosis': This technical functionality diagnoses various fluctuations in the work center based on status information. A diagnosis at the work-center-level is not about quality defects and equipment failures; instead, it involves the observation of an abnormal situation diagnosed in the AAS at a lower level or the prediction of a result for an abnormal situation. Further, it provides a diagnosis or prior diagnosis about whether the specified production plan can be executed normally. This technical functionality is used in the operation execution phase, and it must have the information objects to execute prognostic simulation.
VI. 'Dynamic and proactive response': This technical functionality responds dynamically or in advance in a scenario that requires a response to the present problem or a predicted problem in the work center. This technical functionality—used in the operation execution phase—requires the horizontal coordination of applications so that it can respectively respond or proactively respond to various scenarios diagnosed or predicted through the 'diagnosis and proactive diagnosis' technical functionality. Further, this technical functionality requires the property to support the horizontal coordination and execute a reactive simulation.

Architectural framework for DT-based technical functionality with VREDI

Figure 2 shows the architectural framework that provides six technical functionalities by creating, synchronizing, and utilizing a DT based on VREDI. This architectural framework is designed based on the AAS concept considering the SOA principle with high-level interoperability and efficient information management. Because VREDI is an asset description used in a DT application in work-center-level AAS, it can be used only when there is a DT application. The component manager obtains information objects by approaching the virtual representation or technical functionality of one's own or another AAS and a DT application, which operates the DT on the basis of the information. VREDI complies with the properties of RAMI 4.0 and SOA; thus, many engineering applications are interconnected based on the component manager, thereby forming a loosely-coupled integration. The DT application has a DT engine, an operation module, and a CDL. The major functions of this application include DT creation, synchronization, and utilization based on the DT.

The engineering application provides the technical functionality required for the horizontal coordination of DT-based technical functionalities. These applications contain a service host and core functional engines. The core functional engines refer to the essential engine, rule, algorithm, etc. for each technical functionality. The required core functional engines of the technical functionalities in the type stage are given below:

- 'Resource allocation': When this technical functionality is provided, it must include the core functional engine, which contains rules that allocate required manufacturing resources based on the characteristics of the product to be produced and the process involved.

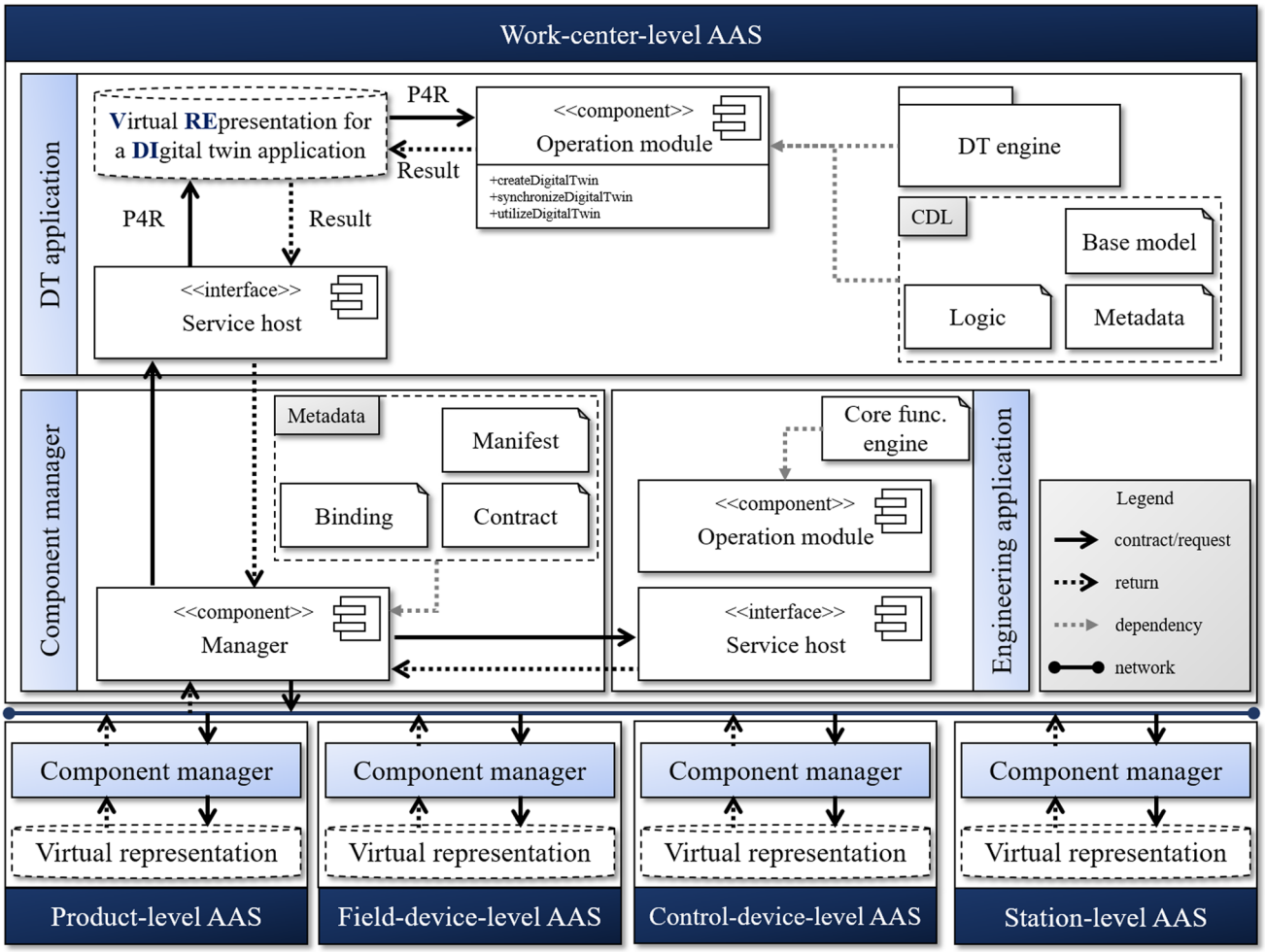

Fig. 2 Architectural framework for DT-based technical functionality

- 'Factory logistics design': This technical functionality must be created in connection with an application that has rules for designing the layout and logistics of the field based on the specified number of production resources.

The required core functional engines of the technical functionalities in the instance stage are provided below:

- 'Planning and scheduling': This technical functionality must be connected to an application that derives the production plan and sequence by date within the capacity of the field. In this case, heuristic rules or metaheuristics, optimization algorithms, and simulation-based optimization algorithms can be used as the core functional engines of the application.
- 'Time-machine monitoring': This technical functionality requires metadata and rules for re-aggregating VREDI in accordance with the time and object desired by the user.
- 'Diagnosis and proactive diagnosis': This technical functionality must have a rule for recognizing the scenario; this rule should be used in conjunction with the simulation result of the DT, to which RBR or case-based reasoning, a classification technique, and a clustering method can be applied.
- 'Dynamic and proactive response': This technical functionality requires the horizontal coordination of applications so that it can either reactively or proactively respond to each of the various scenarios diagnosed or predicted through the 'diagnosis and proactive diagnosis' technical functionality. In this case, heuristic rules or metaheuristics, optimization algorithms, and simulation-based optimization algorithms can be used as the core functional engines of the application.

VREDI structure and relations

VREDI has an object-oriented design that satisfies the various constraints of the AAS in RAMI 4.0. In addition, it contains various elements comprising the work center, the relationships between these elements, the performed

processes, and various plans for production. VREDI consists of five classes, denoted as P4R. Among them, *Product, Process*, and *Resource* are Type classes, and *Plan* and *Plant* are Instance classes in which the defined types are converted to instances. This is the main characteristic of the AAS in RAMI 4.0, and it has the advantage of minimizing duplicated information.

Figure 3 shows a schematic diagram of the elements of each category. This diagram formalizes the various elements in the physical work center and the relationships between them for clarity. These classes are defined as follows:

- *Product* This class includes information about the product to be provided to the entities in the customer or supply chain. The main elements include the BOM, the parts comprising the BOM, and pertinent information about the customers who will receive the final manufactured product sold or delivered to them.
- *Plan* This class includes information about the plans that are provided to or received from the entities in the customer or supply chain and the production implementation plan. The main elements of this information are the procurement plan for receiving materials for production, target production quantity, and work schedule.
- *Process* This class includes processes in the relationships among products, facilities, and MHE. The main elements are processes, material handling operation, and inputs.
- *Resource* This class includes information about the various manufacturing resources required for performing processes and producing products. The main elements are information and definitions of manufacturing resources such as machines, MHE, and buffers. The MHE can be categorized into MHC, MHR, and MHV.
- *Plant* This class is used to convert type information from the *Product*, *Process*, and *Resource* classes into instances, thereby creating and synchronizing the DT. This class includes elements for synchronizing the layout, location, and IoT. The *Product* class does not directly reference this class; however, the *Plant* class refers to the

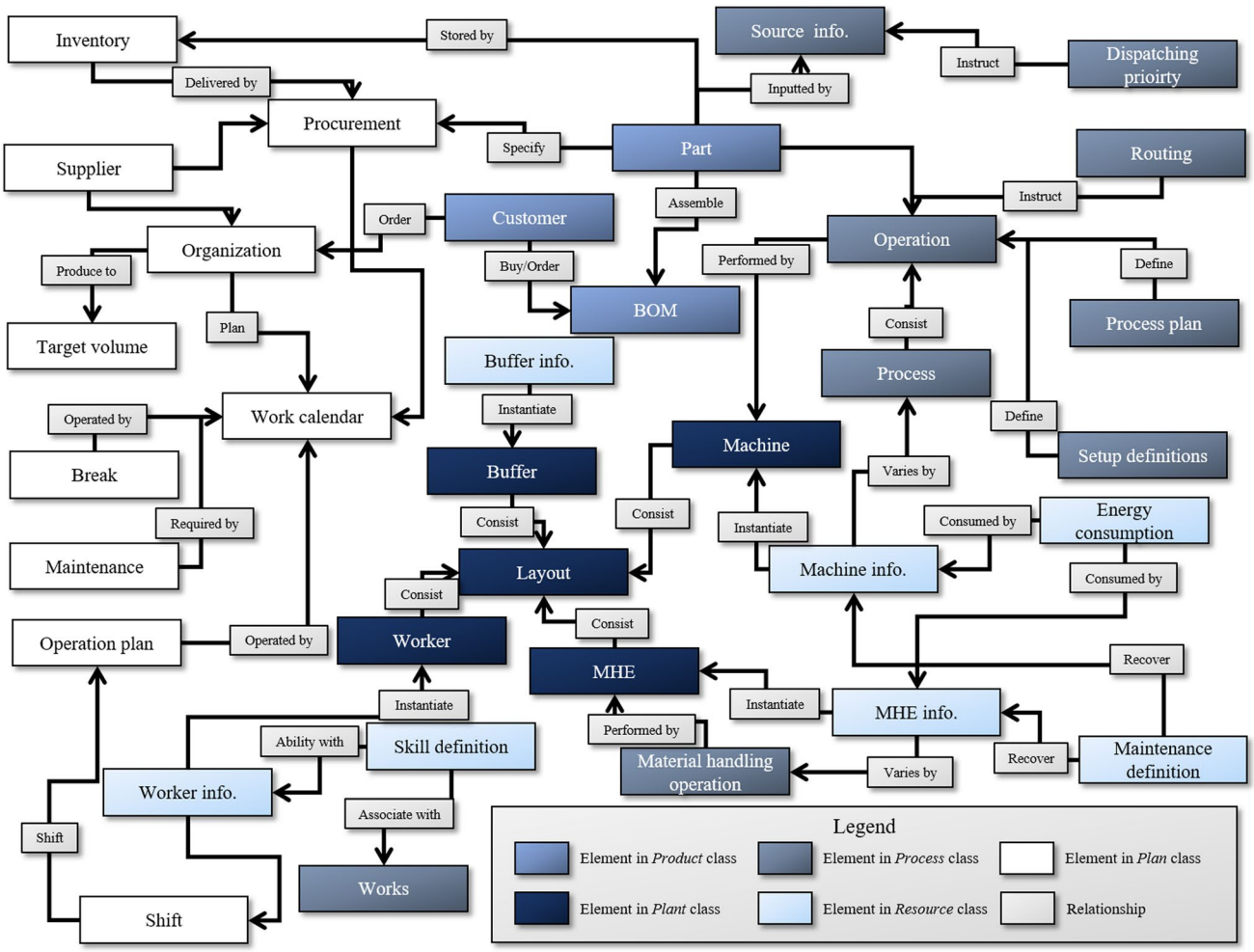

Fig. 3 Schematic diagram of VREDI

Process class that references the *Product* class, and thus, it can use the instance information of the *Product* class.

The relationships among the elements that contain these various elements in VREDI are defined in Fig. 4. All elements form reference relations except the area element, which contains environment information such as the space, aisle, and walls of the work center. The most central class is the *Product* class, which is referenced by the *Process, Plan*, and *Plant* classes. All elements attempt to improve efficiency in terms of total information management by applying the object-oriented and "type and instance" concepts. Thus, information is not simply listed but mapped through mutual references.

Detailed descriptions of VREDI

VREDI must access other AASs or components and receive information for operation procedure of DT application. Furthermore, a Header that describes the location of information is required to create the technical functionalities of applications based on the component manager. The information contained in the Header of VREDI is shown in Table 3. It includes the ID of the AAS, the assets of the AAS, the classes, and the element. As a property of AAS, when a component manager sends or receives data based on a manifest, the Header takes the role of metadata so that it can receive the data even without knowing the direct path. 'RegisteredTime' indicates the time when each class is registered in the AAS and is used to perform technical functionality at the respective time.

The actual data of VREDI, i.e., the five classes of the P4R structure in the body of the AAS that contains information about the respective asset are independent; however, the classes can form relationships with each other through references. This reflects the characteristics of the AAS of RAMI 4.0, which allows distributed storage and mutual referencing. Furthermore, the elements of the classes can be provided from the AAS or engineering application of other levels, and each class can comprise multiple elements. Tables 3, 4, 5, 6 and 7 list the lowest level and cardinality for the AAS of a different level from that of each element. The container is an attribute that does not contain an actual information object; however, it provides an explanation of the group of elements. The manifest is the metadata, which is the one of the main elements in the AAS property, and it provides an address for aggregating the information object to the component manager. The manifest of the lowest level is designed to exploit the advantages of RAMI 4.0, which can use a reference to minimize the data duplication problem. When information is referenced according to the manifest in this manner, the reference information has the advantage that it reflects the technical functionalities created by applications in the AAS.

The elements of the *Product* class are summarized in Table 4.

- 'ProductInfo': An element of the abstracted structures of the product information and BOM below it. The subpart, subassemblies, and final product are all referred to as 'Part,' and this Part has a BOM level and unit quantity. The final product, which is the top 'Part,' does not have a 'ParentPartID.' The 'CustomerID' is referenced by the work-center-level AAS to effectively receive technical functionalities related to planning and scheduling. Thus, the information of the AAS that reflects the various needs of the customers can be retrieved by referencing the product-level AAS.

The *Resource* class has six top elements as summarized in Table 5. All elements included in this class have the Type information. In the 'MachineInfo' and 'WorkerInfo' containers, multiple types in 'TypeInfo' are used to represent machines, workbenches, or workers who can perform multiple process types. All information except for the information related to workers is referenced by the field-device-level AAS.

- 'MachineInfo': A container that contains attributes of the machines or workbenches for every discrete process operation. In work-center-level technical functionality, a simulation is needed for the information around discrete operations.
- 'BufferInfo': An element that contains attributes handling all intermediate storage spaces.
- "MHCInfo": An element that represents a conveyor-type MHE for the material handling operation that transports parts, sub-assemblies, and products through the static position and linear resource operation.
- "MHRInfo": An element that represents a robot-type MHE for the material handling operation that transports parts, sub-assemblies, and products through the static position and robotics operation.
- "MHVInfo": An element that represents a vehicle-type MHE for the material handling operation that transports parts, sub-assemblies, and products through the dynamic position and operation using defined roads or rails.
- 'WorkerInfo': An element that represents a worker type who performs the process operation.
- 'MHVChargerInfo': An element that represents the charger of the MHV.

The *Process* class has three top elements as listed in Table 6. All elements included in this class have the Type information.

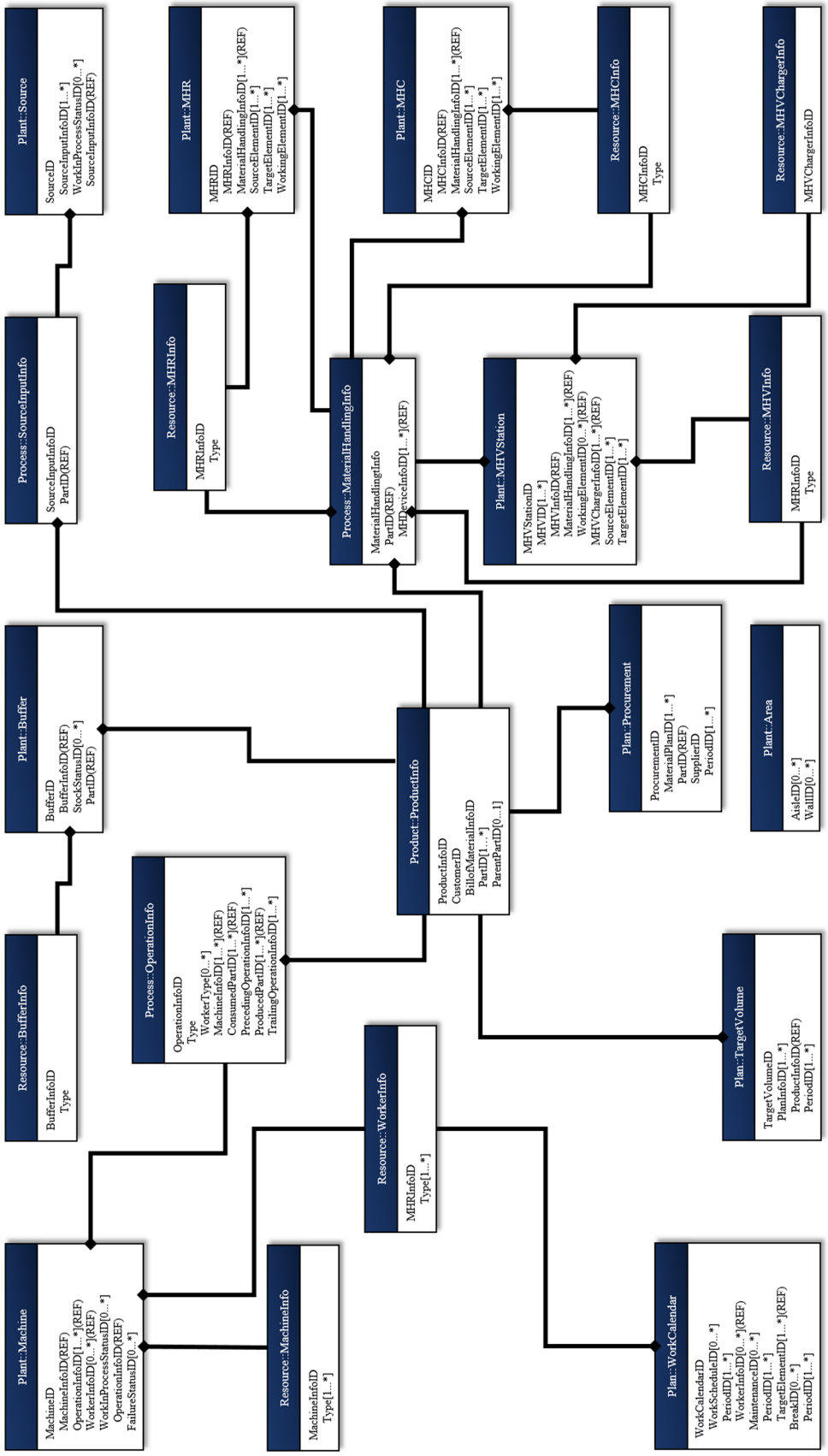

Fig. 4 Identifier relation of elements in the object-oriented structure

Table 3 Header of VREDI

Lvl. 0	Lvl. 1	Lvl. 2	Cardinality	Description
AssetAdministrationShellID				ID of the AAS
	WorkCenterID			ID of the selected work center in the information object
	ClassID		0…*	Container of the selected class in the information object
		TopElementID	0…*	ID of the selected element in the information object
		RegisteredTime		The registered time for aggregation of the information object
	UnitDescription			Unit measures for the information object
	TechnicalFunctionality			Required technical functionality for the DT application

Table 4 Product class of VREDI

Lvl 0	Lvl 1	Lvl 2	Lvl 3	Manifest	Cardinality	Description
ProductInfo				<container>	0…*	Element of product information that is the desired output of the work center
	ProductInfoID			Product		ID of the product
	CustomerID			Work center		ID of a customer placing an order
	BillOfMaterialsInfo			<container>		Container of the BOM information of the product
		BillOfMaterialsInfoID		Product		ID of BOM of the product
		Part		<container>	1…*	Container of the part/subassembly in the BOM of the product
			PartID	Product		ID of the part/subassembly in the BOM of the product
			BOMLevel	Product		Level of the part/subassembly in the BOM tree
			PartUnitNum	Product		The number of required units of the part/subassembly for the parent part/subassembly
			ParentPartID	(REF)	0…1	Reference ID of the parent part/subassembly

- 'OperationInfo': An element referring to the process operation. This container considers the type for independent processes and the three dimensions of the operation: Product, Process, and Resource. Further, it considers only discrete operations where the parts in the 'PartConsumed' container are converted to the parts in 'PartProduced.' Meanwhile, if various 'MachineInfoIDs' are referenced, the processing time for each machine may differ; however, they will have the same input and output for the process. In addition, the 'PartIDs' are referenced for managing information object based on each part of the operation. Meanwhile, operation and job-related information are received from the control-device-level AAS to provide the recipe information of applications such as process control. Furthermore, the information of the 'PartConsumed' and 'PartProduced' containers are received from the product-level AAS to provide technical functionalities such as process planning.
- 'SourceInputInfo': An element referring to input operations. The input order and cycle can be received in planning and scheduling or proactive and dynamic responses. The information of this element is not received from the AAS of another level but rather saved or created in the work-center-level AAS.
- 'MaterialHandlingInfo': An element referring to all types of transport operations. The information inside this container is about logistics in the work center, and it is handled by a station-level or higher level AAS because the information is not about a single piece of equipment.

The *Plan* class has three top elements as listed in Table 7. All elements included in this class have Instance information. This container has information about the plans of the entire work. Thus, the main container has the work-center-level AAS as the manifest.

- 'WorkCalendar': An element of the plan of the operation, maintenance, and break of the work center. This element is an abstraction of organizing a calendar of uptime, maintenance, and outages for each day. The information in the 'Maintenance' container is received from the field-device-level AAS, and information created based on technical functionalities such as preventive maintenance or predictive maintenance can be used.
- 'Procurement': An element of the procurement plan from other subjects in the value chain. It is designed to express procurement plans that change over time during the procurement of the same product.

Table 5 *Resource* class of VREDI

Lvl 0	Lvl 1	Lvl 2	Lvl 3	Manifest	Cardinality	Description
MachineInfo				<container>	0…*	Element of the machine property and information for coordinating with logic in the CDL
	MachineInfoID			Field device		ID of the machine type, which is information for coordination with logic in the CDL
	Property			<container>		Container of the machine property
		Capacity		Field device		Ability to perform the production process in parallel
		TypeInfo		<container>	1…*	Container of process type that can be performed in the machine
			Type	Field device		Process type that can be performed in the machine type
			Priority	Field device		Priority when multiple process types can be performed
		MeanTimeBetweenFailure		Field device		Expected MTBF value of the machine type
		MeanTimeToRepair		Field device		Expected MTTR value of the machine type
		FailureRate		Field device		Expected failure rate of the machine type
		OperationConsumption		Field device		Energy consumption in the operation of the machine type
		IdleConsumption		Field device		Energy consumption in the idle time of the machine type
		Size		Field device		Designed size of the machine type
BufferInfo				<container>	0…*	Element of buffer property and information for coordinating with logic in the CDL
	BufferInfoID			Field device		ID of the buffer type, which is information for coordination with the logic in the CDL
	Property			<container>		Container of the buffer property
		Capacity		Field device		Ability to hold the inventory/WIP
		Type		Field device		Operation type that can be performed in the buffer type
		Size		Field device		Designed size of the buffer type
MHCInfo				<container>	0…*	Element of the MHC property and information for coordinating with logic in the CDL
	MHCInfoID			Field device		ID of the MHC type, which is information for coordination with the logic in the CDL
	Property			<container>		Container of the MHC property
		Type		Field device		Material handling process type that can be performed in the MHC type
		MeanTimeBetweenFailure		Field device		Expected MTBF value of the MHC type
		MeanTimeToRepair		Field device		Expected MTTR value of the MHC type
		FailureRate		Field device		Expected failure rate of the MHC type
		OperationConsumption		Field device		Energy consumption in the operation of the MHC type
		IdleConsumption		Field device		Energy consumption in the idle time of the MHC type
		TransportationSpeed		Field device		Transportation speed of the MHC type
		Width		Field device		Designed width of the MHC type
MHRInfo				<container>	0…*	Element of MHR property and information for coordinating with the logic in the CDL
	MHRInfoID			Field device		ID of the MHR type, which is information for coordination with logic in the CDL
	Property			<container>		Container of MHR property
		Type		Field device		Material handling process type that can be performed in the MHR type

Table 5 (continued)

Lvl 0	Lvl 1	Lvl 2	Lvl 3	Manifest	Cardinality	Description
		MeanTimeBetweenFailure		Field device		Expected MTBF value of the MHR type
		MeanTimeToRepair		Field device		Expected MTTR value of the MHR type
		FailureRate		Field device		Expected failure rate of the MHR type
		OperationConsumption		Field device		Energy consumption in the operation of the MHR type
		IdleConsumption		Field device		Energy consumption in the idle time of the MHR type
		RoboticsConfiguration		Field device		Robotics configuration setting of the MHR type
		Size		Field device		Designed size of the MHR type
MHVInfo				<container>	0…*	Element of the MHV property and the information for coordinating with logic in the CDL
	MHVInfoID			Field device		ID of the MHV type, which is information for coordination with logic in the CDL
	Property			<container>		Container of the MHV property
		Type		Field device		Material handling process type that can be performed in the MHV type
		MeanTimeBetweenFailure		Field device		Expected MTBF value of the MHV type
		MeanTimeToRepair		Field device		Expected MTTR value of the MHV type
		FailureRate		Field device		Expected failure rate of the MHV type
		BatteryCapacity		Field device		Battery capacity of the MHV type
		RechargeQuantity		Field device		Battery recharge quantity in the unit time of the MHV type
		OperationConsumption		Field device		Energy consumption in the operation of the MHV type
		IdleConsumption		Field device		Energy consumption in the idle time of the MHV type
		TurnSpeed		Field device		Turn speed of the MHV type
		OperationSpeed		Field device		Operation speed of the MHV type
		TransportationSpeed		Field device		Transportation speed of the MHV type
		Size		Field device		Designed size of the MHV type
WorkerInfo				<container>	0…*	Element of worker property and information for coordinating with the logic in the CDL
	WorkerInfoID			Work center		ID of the worker type, which is information for coordination with the logic in the CDL
	Property			<container>		Container of worker property
		TypeInfo		<container>	1…*	Container of process type that can be performed by the worker type
			Type	Work center		Process type that can be performed by the worker type
			Skill	Work center		Proficiency of the process type based on the worker type
MHVChargerInfo				<container>	0…*	Element of the MHV charger property and information for coordinating with the logic in the CDL
	MHVChargerInfoID			Field device		ID of the MHV charger type, which is information for coordination with the logic in the CDL
	Property			<container>		Container of the MHV charger property
		Capacity		Field device		Ability to charge the MHV in parallel
		Size		Field device		Designed size of the MHV charger type

Table 6 *Process* class of VREDI

Lvl 0	Lvl 1	Lvl 2	Manifest	Cardinality	Description
OperationInfo			<container>	0…*	Element of information on a process operation
	OperationInfoID		Control device		ID of the process operation information
	Type		Control device		Operation type performed in resources
	WorkInfo		<container>	0…*	Container of the required work information
		Type	Work center		Required work type for the process operation
		WorkerNum	Work center		Required worker number
	ProcessInfo		<container>	1…*	Container of process by the resource type in the process operation
		MachineInfoID	(REF)		Reference ID of the resource type
		ProcessingTime	Control device		Processing time of the process in the resource type
		SetupTime	Control device		Setup time of the process in the resource type
		PoorQualityRate	Control device		Poor quality rate of the process in the resource type
		ReworkRate	Control device		Rework rate of the process in the resource type
	PartConsumed		<container>	1…*	Container of information input to the process operation
		PartID	(REF)		Reference ID of the part that is input to the process operation
		PrecedingOperationInfoID	Product		ID of the preceding process operation
		PartNum	Product		The number of inputs of the part in the process operation
	PartProduced		<container>	1…*	Container of the output information of the process operation
		PartID	(REF)		Reference ID of the part that is output from the process operation
		TrailingOperationInfoID	Product	0…1	ID of the trailing process operation
		PartNum	Product		The number of outputs of the part in the process operation
SourceInputInfo			<container>	0…*	Element of information on a source (input operation)
	SourceInputInfoID		Work center		ID of the source
	PartID		(REF)		Reference ID of the part that is input from the source
	DispatchingPriority		Work center/application		Dispatching priority of the part
	PitchTime		Work center/application		Pitch time by the part

Table 6 (continued)

Lvl 0	Lvl 1	Lvl 2	Manifest	Cardinality	Description
MaterialHandlingInfo			<container>	0…*	Element of information on a material handling operation
	MaterialHandlingInfoID		Station		ID of the material handling operation
	PartID		(REF)		Reference ID of the part that is the material for transportation
	LotSize		Station		Unit number in a lot for transportation
	MaterialHandlingProcessInfo		<container>	1…*	Container of material handling operation by resource type
		MHEInfoID	(REF)		Reference ID of the MHE type
		PushPull	Station		Boolean value for indicating push/pull of the material handling operation
		LoadingTime	Station		Loading time of the material handling operation
		UnloadingTime	Station		Unloading time of the material handling operation

- 'TargetVolume': An element indicating the target production volume. It is designed to express the target production volumes that change over time during the production of the same product. These containers can be coordinated with planning and scheduling and proactive and dynamic response functionalities, which are created based on the application of the work-center-level AAS. The 'ProductionStatus' can reflect the current production scenario of the number of products already produced in the time point where DT is synchronized.

The *Plant* class is a class used to convert the type information from *Product*, *Process*, and *Resource* classes into instances, as listed in Table 8. For this class, information is collected based on the current field information in the instance stage of the work center. Further, the current field information enables vertical integration for simulation with synchronization in the physical asset. The top elements are containers in which the manufacturing resources of the *Resource* class are converted to instances. The number of instances such as 'Machine,' 'Buffer,' and 'Worker' can be determined based on the resource allocation technical functionality, and the instances of the 'MHVStation,' 'MHC,' and 'MHR' can be created based on the factory logistics design technical functionality. Finally, the 'Location' information inside all containers can be created based on the factory logistics design.

- 'Machine': An element in which 'MachineInfo,' 'OperationInfo,' and 'WorkerInfo' elements are converted to instances. This element handles information such as the WIP, machine failure, and confidence factor from the control-device-level AAS, and the information is used in the 'diagnosis and proactive diagnosis' technical functionality. Meanwhile, the control-device-level AAS can upgrade the 'diagnosis' technical functionality by predicting anomalies based on quality prediction and prognostic health management technical functionalities.
- 'Source': An element in which the 'SourceInputInfo' element is converted to an instance. This element contains the current WIP information in the source in the 'WorkInProcessStatus' container.
- 'Buffer': An element in which the 'BufferInfo' element is converted to an instance. This element indicates the current buffer status in the 'StockStatus' container.
- 'MHVStation': An element in which the 'MHVInfo,' 'MHVChargerInfo,' and 'MaterialHandlingInfo' elements are converted to instances. This element indicates the current material handling operation, current MHV status, and instance information of the MHV.
- 'MHC': An element in which the 'MHCInfo' and 'MaterialHandlingInfo' elements are converted to instances. This element contains the current MHC status and instance information of the MHC.

Table 7 *Plan* class of VREDI

Lvl 0	Lvl 1	Lvl 2	Lvl 3	Manifest	Cardinality	Description
WorkCalendar				<container>		Element of the operation, maintenance, and break plan at the work center
	WorkCalendarID			Work center		ID of the operation, maintenance, and break plan
	WorkSchedule			<container>	0…*	Container of the work center operation plan
		WorkScheduleID		Work center		ID of the work center operation plan
		Period		<container>	1…*	Container of period of the work center operation plan
			PeriodID	Work center/application		ID of the period of the work center operation plan
			StartDateTime	Work center/application		Start date and time of the period of the work center operation plan
			EndDateTime	Work center/application		End date and time of the period of the work center operation plan
		Shift		<container>	0…*	Container of the shift information of the worker
			WorkerInfoID	(REF)		Worker in operation in the period of the shift
	Maintenance			<container>	0…*	Container of the work center maintenance plan
		MaintenanceID		Field device		ID of the work center maintenance plan
		Period		<container>	1…*	Container of the period of the work center maintenance plan
			PeriodID	Field device		ID of the period of the work maintenance plan
			StartDateTime	Field device		Start date and time of the period of the work center maintenance plan
			EndDateTime	Field device		End date and time of the period of the work center maintenance plan
		TargetElement		<container>		Container of the target element that participates in the maintenance
			ElementID	(REF)	1…*	Reference ID of the target element that participates in the maintenance
	Break			<container>	0…*	Container of the work center break plan
		BreakID		Work center		ID of the work center break plan
		Period		<container>	1…*	Container of the period of the work center break plan
			PeriodID	Work center		ID of the period of the work center break plan
			StartDateTime	Work center		Start date and time of the period of the work center maintenance plan

Table 7 (continued)

Lvl 0	Lvl 1	Lvl 2	Lvl 3	Manifest	Cardinality	Description
			EndDateTime	Work center		End date and time of the period of the work center maintenance plan
Procurement				<container>		Element of the procurement plan
	ProcurementID			Work center		ID of the procurement plan for the work center
	MaterialPlan			<container>	1…*	Container of the material plan by each part
		MaterialPlanID		Work center		ID of the material plan
		PartID		(REF)		Reference ID of the part in the material plan
		SupplierID		Work center		ID of the supplier in the material plan
		Inventory		Work center		The number of the current inventory of the part
		Period		<container>	1…*	Container of the period of the procurement plan
			PeriodID	Work center		ID of the period of the work center material plan
			ArrivalDateTime	Work center		Arrival date and time of the period of the work center break plan
			ArrivalVolume	Work center		Number of parts that arrived during the period
TargetVolume				<container>		Element of the target production volume
	TargetVolumeID			Work center		ID of the target production volume
	PlanInfo			<container>	1…*	Container of the detailed target production volume plan
		PlanInfoID		Work center/application		ID of the detailed target production volume plan
		Period		<container>	1…*	Container of the period of the detailed target production volume plan
			PeriodID	Work center/application		ID of the period of the detailed target production volume plan
			StartDateTime	Work center/application		Start date and time of the period of the detailed target production volume plan
			EndDateTime	Work center/application		End date and time of the period of the detailed target production volume plan
			ProductionVolume	Work center/application		Target production volume during the period
		ProductInfoID		(REF)		Reference ID of the production in the detailed target production volume plan
		TargetProductionVolume		Work center		Requested target production volume by product
		ProductionStatus		Work center		Current production status of products under production

Table 8 *Plant* class of VREDI

Lvl 0	Lvl 1	Lvl 2	Lvl 3	Manifest	Cardinality	Description
Machine				<container>	0…*	Element of the machine instance that contains the property and current situation
	MachineID			Work center		ID of the machine instance
	MachineInfoID			(REF)		Reference ID of the machine type
	ProcessProperty			<container>		Container of the allocated operation type
		OperationInfoID		(REF)	1…*	Reference ID of the operation type
	WorkerProperty			<container>	0…*	Container of the allocated worker type
		WorkerInfoID		Work center		Reference ID of the worker type
		WorkerNum		Work center/application		Number of allocated workers by worker type
	Location			Work center/application		Physical location of the machine instance
	Transformation			Work center		Physical transformation of the machine instance
	ConfidenceFactor			Control device		Confidence factor of the machine instance
	WorkInProcessStatus			<container>	0…*	Container of the current WIP information in the machine instance
		WorkInProcessStatusID		Control device		ID of the current WIP information in the machine instance
		OperationInfoID		(REF)		Reference ID of the operation type that is performed in the current WIP
		ProcessedTime		Control device		Time elapsed in the operation
		PoorQuality		Control device		Detected or predicted poor quality of the current WIP
	FailureStatus			<container>	0…*	Container of the current machine failure information of the machine instance
		FailureStatusID		Control device		ID of the current machine failure information
		Type		Control device		Machine failure type in the current machine failure information
		RepairedTime		Control device		Elapsed repair time
Source				<container>	0…*	Element of the instance for input operations that contains the property and current situation

Table 8 (continued)

Lvl 0	Lvl 1	Lvl 2	Lvl 3	Manifest	Cardinality	Description
	SourceID			Work center		ID of the instance for input operations
	SourceProperty			<container>		Container of the allocated source type in the instance
		SourceInputInfoID		(REF)	1…*	Reference ID of the information on a source type
	Location			Work center/application		Physical location of the instance for input operations
	Transformation			Work center		Physical transformation of the instance for input operations
	WorkInProcessStatus			<container>	0…*	Container of the current WIP information in the instance for input operations
		WorkInProcessStatusID		Control device		ID of the current WIP information in the instance for input operations
		SourceInputInfoID		(REF)		Reference ID of the input operation type that is performed in the current WIP
Buffer				<container>	0…*	Element of the buffer instance that contains the property and current situation
	BufferID			Work center		ID of the buffer instance
	BufferInfoID			(REF)		Reference ID of the buffer type
	Location			Work center/application		Physical location of the buffer instance
	Transformation			Work center		Physical transformation of the buffer instance
	StockStatus			<container>	0…*	Container of the current stock information in the buffer instance
		StockStatusID		Control device		ID of the current stock information in the buffer instance
		PartID		(REF)		Reference ID of the part that is stocked in the buffer instance
		StockNum		Control device		Number of stocks that have the part ID in the buffer instance
MHVStation				<container>	0…*	Element of the station with the MHV instance that contains the property and current situation
	MHVStationID			Work center		ID of the station with the MHV instance

Table 8 (continued)

Lvl 0	Lvl 1	Lvl 2	Lvl 3	Manifest	Cardinality	Description
	MHVProperty			<container>	1…*	Container of the MHV instance
		MHVID		Station		ID of the MHV instance
		MHVInfoID		(REF)		Reference ID of the MHV type
		Location		Station		Physical location of the MHV instance
		MaterialHandlingProperty		Station	1…*	Container of the allocated material handling operation type
			MaterialHandling-PropertyID	Station		ID of the allocated material handling operation type
			MaterialHandlingIn-foID	(REF)		Reference ID of the allocated material handling operation type
		MaterialHandlingStatus		<container>	0…1	Container of the current material handling operation information in the MHV instance
			MaterialHandlingSta-tusID	Work center		ID of the current material handling operation information in the MHV instance
			ElementID	(REF)		Reference ID of the instance that is performed in the current material handling operation with the MHV instance
			MaterialHandlingIn-foID	(REF)		Reference ID of the material handling operation that is performed by the MHV instance
	MHVChargerProperty			<container>	1…*	Container of MHV charger instance
		MHVChargerInfoID		(REF)		Reference ID of the MHV charger type
		MHVChargerNum		Station		The number of the MHV charger instance that has the MHV charger type
	SourceElement			<container>		Container of the instance that provides the part for the material handling operation
		ElementID		(REF)/application	1…*	Reference ID of the instance that provides the part for the material handling operation

Table 8 (continued)

Lvl 0	Lvl 1	Lvl 2	Lvl 3	Manifest	Cardinality	Description
	TargetElement			<container>		Container of the instance that receives the part from the material handling operation
		ElementID		(REF)/application	1…*	Reference ID of the instance that receives the part from the material handling operation
	Location			Work center/application		Physical location of the station with the MHV instance
MHC				<container>	0…*	Element of the MHC instance
	MHCID			Work center		ID of the MHC instance
	MHCInfoID			(REF)		Reference ID of the MHC type
	MaterialHandlingProperty			<container>		Container of the allocated material handling operation type
		MaterialHandlingInfoID		(REF)	1…*	Reference ID of the allocated material handling operation type
	SourceElement			<container>		Container of the instance that provides the part for the material handling operation
		ElementID		Work center/application	1…*	Reference ID of the instance that provides the part for the material handling operation
	TargetElement			<container>		Container of the instance that receives the part from the material handling operation
		ElementID		Work center/application	1…*	Reference ID of the instance that receives the part from the material handling operation
	MaterialHandlingStatus			<container>	0…1	Container of the current material handling operation information in the MHC instance
		MaterialHandlingStatusID		Work center		ID of the current material handling operation information in the MHC instance

Table 8 (continued)

Lvl 0	Lvl 1	Lvl 2	Lvl 3	Manifest	Cardinality	Description
		ElementID		(REF)		Reference ID of the instance that is performed in the current material handling operation with the MHC instance
		MaterialHandlingInfoID		(REF)		Reference ID of the material handling operation that is performed by the MHC instance
MHR				<container>	0…*	Element of the MHR instance
	MHRID			Work center		ID of the MHR instance
	MHRInfoID			(REF)		Reference ID of the MHR type
	MaterialHandlingProperty			<container>		Container of the allocated material handling operation type
		MaterialHandlingInfoID		(REF)	1…*	Reference ID of the allocated material handling operation type
	SourceElement			<container>		Container of the instance that provides the part for the material handling operation
		ElementID		Work center/application	1…*	Reference ID of the instance that provides the part for the material handling operation
	TargetElement			<container>		Container of the instance that receives the part from the material handling operation
		ElementID		Work center/application	1…*	Reference ID of the instance that receives the part from the material handling operation
	MaterialHandlingStatus			<container>	0…1	Container of the current material handling operation information in the MHR instance
		MaterialHandlingStatusID		Work center		ID of the current material handling operation information in the MHR instance
		ElementID		(REF)		Reference ID of the instance that is performed in the current material handling operation with the MHR instance

Table 8 (continued)

Lvl 0	Lvl 1	Lvl 2	Lvl 3	Manifest	Cardinality	Description
		MaterialHandlingInfoID		(REF)		Reference ID of the material handling operation that is performed by the MHR instance
Area				<container>		Element of the area of the work center
	Aisle			<container>	0…*	Container of the aisle in the work center
		AisleID		Work center		ID of the aisle in the work center
		WayType		Work center		Unidirectional/bidirectional type of aisle
		Width		Work center		Width of the aisle
		StartPoint		Work center		Starting point of the aisle
		EndPoint		Work center		Ending point of the aisle
	Wall			<container>	0…*	Container of the wall in the work center
		WallID		Work center		ID of the wall in the work center
		Thickness		Work center		Thickness of the wall
		StartPoint		Work center		Starting point of the wall
		EndPoint		Work center		Ending point of the wall
	Space			<container>		Element of the space information
		VerticalLength		Work center		Vertical length of the work center
		HorizontalLength		Work center		Horizontal length of the work center

- 'MHR': An element in which the 'MHRInfo' and 'MaterialHandlingInfo' elements are converted to instances. This element manages the current MHR status and instance information of the MHR.
- 'Area': An element that contains the space information of the work center.

Industrial case studies

Industrial case study overview

VREDI was applied to two actual work centers that are discrete manufacturing systems in terms of their operational characteristics to validate its performance and ensure that all four core technical requirements of VREDI were met. The industrial case studies implemented different DT applications based on the proposed architectural framework in a heterogeneous application and development environment. In addition, the virtual representation object with suitable P4R elements by each work center was aggregated and instantiated in both industrial case studies. The virtual representation objects were implemented based on the XML format, which is suitable for the SOAP protocol. Moreover, the two industrial case studies also contained validations for the core technical requirements of DT definition and improving the existing asset descriptions. The first industrial case study was performed to validate the inheritance of AAS property requirement, and the second was performed to validate the DT-based technical functionality requirement. The details of the objectives and case study designs of the two industrial case studies are summarized as follows:

- *Industrial case study for comparative analysis of information utilization (Sect. 4.2)* A vehicle component manufacturing system that produces automotive door trims was selected in this case study. This case study was performed to validate the information management aspect.

The proposed VREDI can prevent the use of duplicate information for efficient information management based on the inheritance of the AAS property. In particular, the "type and instance" concept, which is different from the existing asset descriptions and is a property of the AAS, can reduce duplication. For validation, VREDI is compared with NESIS and CMSD, which are existing asset descriptions for a virtual factory.

- *Industrial case study for supporting DT-based technical functionality (Sect. 4.3)* The second case study is a modular manufacturing system for personalized production in an MSF. In addition, with the advanced engineering applications, the second case study validates the fulfillment of the technical requirement related to DT-based technical functionalities. The interconnection with the field based on vertical integration and horizontal coordination characteristics, and the DT-based technical functionality were verified in the MSF. The DT-based technical functionality refers to the six selected technical functionalities in work-center-level AAS. For validation, the industrial case study was performed by developing advanced engineering applications to enable the realization of the six DT-based technical functionalities.

Industrial case study for comparative analysis of information utilization

Application environment

The vehicle component manufacturing system is a discrete manufacturing system wherein workers assemble parts on workbenches. The actual view of the work center is shown in Fig. 5. The work center produces door trims, which are parts supplied to a vehicle manufacturer. The subparts are supplied from other subjects in the value chain, and this work center only performs the assembly and thermosetting processes.

The layout of the vehicle component manufacturing system is shown in Fig. 6. At this work center, the logistics is simply an assembly line flowing through the conveyor system. All processes consist of assembly and thermosetting, except for the weather-stripping process. Although the process types are identical, they consist of different stations because the front and rear doors, and the left and right doors have different shapes owing to the characteristic of the door trims. A thermosetting machine is used for the thermosetting process after assembly. An assembly workbench is used by workers to assemble door trims and parts. An inspection workbench is used by workers to inspect door trims and exists at the end of the line. A conveyor moves across each machine or workbench, and it is the only logistic device. The weather-stripping machine is used in the weather-stripping process and has several types of racks, which are applied according to the application. This vehicle component manufacturing system has five lines, which are identical to the lines in Fig. 6.

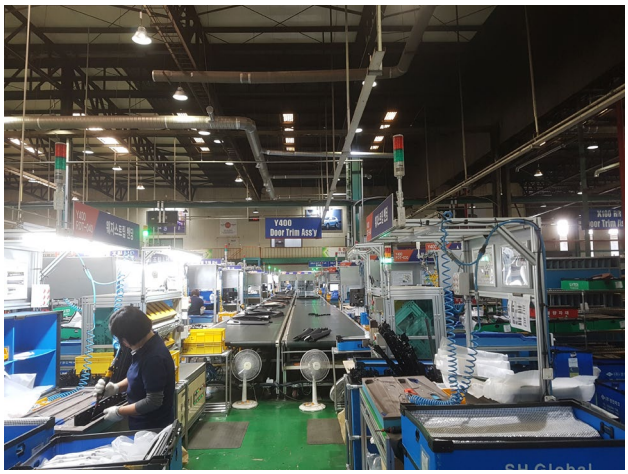

Fig. 5 Vehicle component manufacturing system in Iksan, Republic of Korea

Application information

Table 9 summarizes the application information of the experiment performed for the vehicle component manufacturing system. For this case study, the interface of the operation module of the DT application is configured to parse the information object based on CMSD and NESIS as well as the proposed VREDI, for creating and synchronizing a virtual factory with simulation as its core technical functionality. These two asset descriptions were selected for comparison because they have detailed descriptive examples that minimize the possibility of misuse. Because the two asset descriptions do not contain elements for vertical integration with the site, the case study is designed to exclude these elements to ensure consistent condition in the experiment. The validation of the elements related to vertical integration is performed in the case study described in Sect. 4.3. The detailed descriptions of the application information are as follows:

- *Component manager and service host* These components were implemented using a WCF framework. The WCF framework is suitable for the SOA principle, which requires loosely-coupled integration. In this framework, the SOAP was applied to the component manager as it can be used in a stable manner. The manifest, binding, and contract information was prepared in XML format for universal use and practical application.
- *DT application* This application was implemented based on the proposed architectural framework. The DT engine

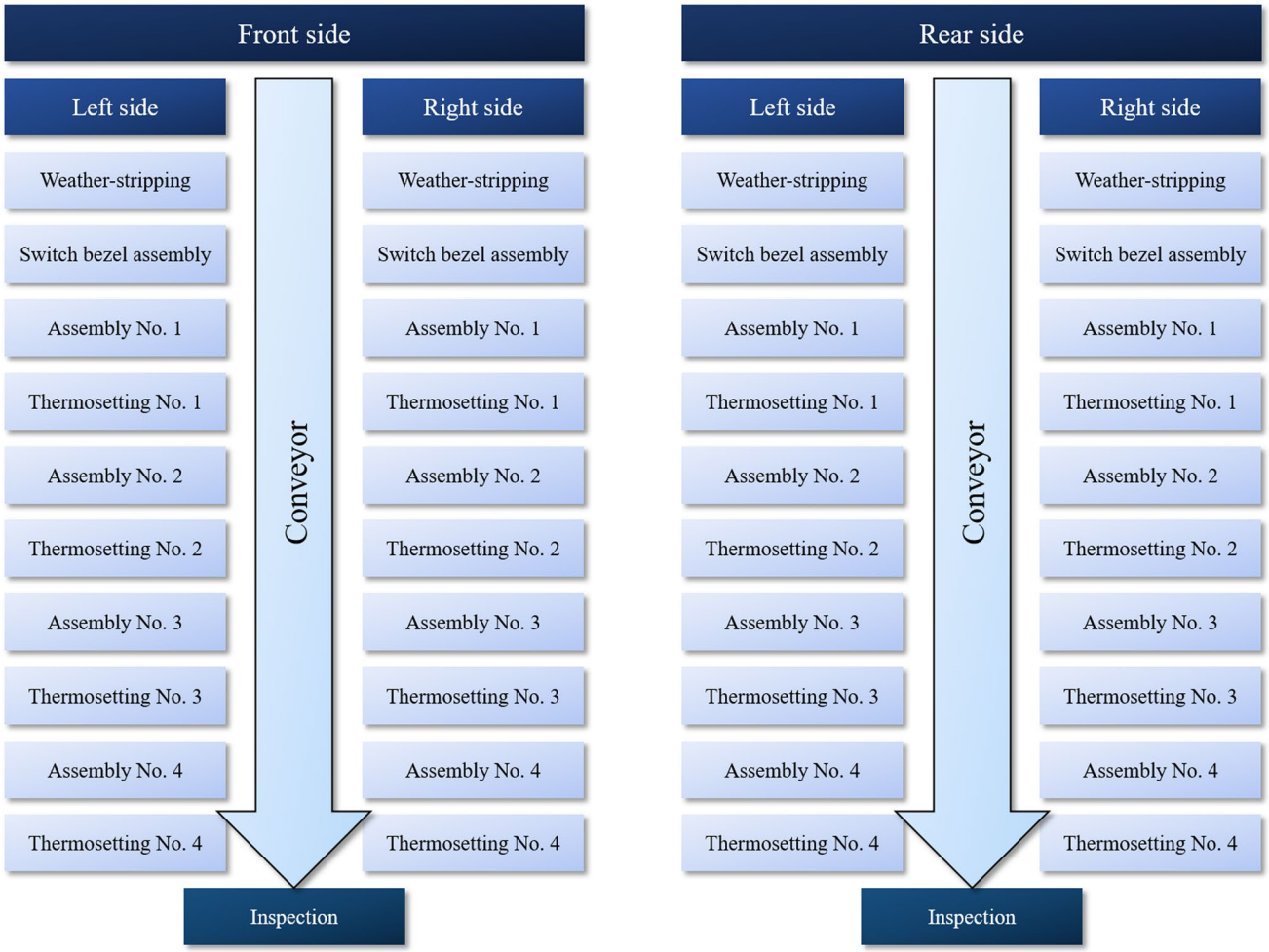

Fig. 6 Layout of the vehicle component manufacturing system

of the DT application was selected to apply Visual Components to the vehicle component manufacturing system. This DT engine supports DES and is suitable for the vehicle component manufacturing system, which is a discrete manufacturing system. Additionally, the DT engine provides a sufficient API library with the C# assembly language for efficient development. The operation module of the DT application was implemented in the C# assembly language as it is compatible with the API library of the DT engine. Figure 7 shows the base models of the DT application for the vehicle component manufacturing system. The base models in CDL are implemented to enable universal use according to the STEP standard. Additionally, the logic and metadata in CDL are implemented in XML format for universal use.

DT application for comparative analysis of information utilization

The components in this case study are implemented to verify that the proposed VREDI can prevent the usage of duplicate information for efficient information management based on the inheritance of the AAS property. Figure 8 shows the implemented DT application and the virtual factory created and synchronized based on VREDI. Figure 8a shows the visual components, which comprise the DT engine. Figure 8b shows the base model imported based on the metadata; Fig. 8c shows the logic and synchronized information; and Fig. 8d shows the created and synchronized virtual factory of the vehicle component manufacturing system. In this experiment, the work center

Table 9 Application information of components for comparative analysis

Component	Item	Contents
DT application	Development environment	Visual Studio 2015
	Programming language	C#
	Programming framework	.NET Framework 4.6.2
	DT engine	Visual Components 4.1
	Base model	STEP
	Metadata	XML
	Logic	XML
Component manager	Development environment	Visual Studio 2015
	Programming language	C#
	Programming model	WCF
	Programming framework	.NET Framework 4.6.2
	Manifest	XML
	Binding	XML
	Contract	XML

was also described with NESIS and CMSD, and other virtual factories were created and synchronized by NESIS and CMSD.

The number of information objects was compared instead of the capacity of the information object because the information object capacity for some attributes such as ID was meaningless, and therefore, these objects were excluded. Furthermore, VREDI does not have separate description elements because it only contains data to be used in the DT rather than the entire information system. For proper testing, all user-oriented descriptions not used in the simulation (e.g., unit description) were excluded. Furthermore, to verify the duplication of unnecessary information, the amount of information used uniquely as an asset description of the virtual factory was also compared. Table 10 summarizes the results of a comparison of VREDI with NESIS and CMSD. The amount of information and the amount of unique information of VREDI used to establish the DT of the vehicle component manufacturing system were set as 100.00%, and this was compared with the cases where other asset descriptions were used. Each information object utilized by the asset description was counted to extract the number of information objects. Additionally, any unique information objects utilized were counted, whose type of information under the "type and instance" concept was not duplicated. Finally, the number of information objects without duplication was derived by dividing the utilized unique information objects by the total number of information objects.

As shown in Table 10, the proposed VREDI can better prevent the use of duplicated information compared to other existing asset descriptions. In particular, the "type and instance" concept, which is one of the properties of the AAS, helps reduce the duplication. As an example of information describing the processing time and machine failure of manufacturing resources, the information corresponding to the number of resources that perform the same behavior must be entered if "type and instance" concepts are not applied. As NESIS does not accommodate the "type and instance" concept, it exhibits a significant difference in the case of the vehicle component manufacturing system, which has many identical resources performing identical operations for identical products. Thus, NESIS includes approximately 72.69% more information than VREDI, but the amount of actual unique information is less by approximately 10.92%. Furthermore, NESIS cannot provide detailed descriptions because it has fewer total data elements than VREDI and CMSD.

The CMSD was confirmed to have large amounts of both total information and unique information. However, in terms of unique information object without duplication, the VREDI can provide an information object to a DT application to create and synchronize the DT with less information duplication. The main cause of this result is that, unlike VREDI, CMSD also included job descriptions that are not necessary to create and synchronize the virtual factory at the work-center-level. In contrast, VREDI separated the descriptions for material handling operation, which is important at the work-center-level. In addition, 'PartGroup' in CMSD contained information such as inventory and a large amount of duplicated information in the definitions of relationships between the parts of the product. Furthermore, the information object was duplicated, and there was a larger amount of information similar to the *Plan* class. However, CMSD had fewer differences when compared with NESIS because it separates the descriptions for resource types from the descriptions for the resource instances.

The amount of information referenced by the work-center-level AAS before using VREDI was approximately 62.49% of that of the fully implemented information object of VREDI. This shows that VREDI is efficient in terms of the management of component-manager-enabled aggregation. Furthermore, from the aspect of detailed elements, VREDI also has additional elements. Further, it was confirmed that elements related to energy consumption and elements that affect the operation aspects such as the subdivision of work-center-level operations and delivery have been added. Thus, there are more detailed elements; however, the application of high-level object-oriented design and "type and instance" concepts contribute to efficient information operation.

Fig. 7 Base models for the DT application of the vehicle component manufacturing system

Industrial case study for supporting dt-based technical functionality

Application environment

The MSF, which is the target work center of the second industrial case study, is in Daejeon, Republic of Korea. The actual view of the work center is shown in Fig. 9. This MSF, which is unmanned, produces prototypes and specimens for small-sized enterprises and venture companies using additive manufacturing, various automated machines, and robots. Further, it produces post-processing systems for processing various prototypes and specimens (Park et al. 2019b; Son et al. 2015; Kang et al. 2018).

The layout of the MSF is shown in Fig. 10 (Park et al. 2019b; Kang et al. 2018). It has two stations—a component production station and post-processing station. The MSF utilizes seven production modules, two types of MHE, and one buffer module. The component production station is composed of additive manufacturing modules. The additive manufacturing module, which uses a 3D printer, represents an upgrade from the conventional manufacturing method using molds. It is an effective production method that can respond to wide variety of product requirements. The post-processing station performs processes after the buffer module as well as other modules. The fumigation and polishing module perform post-processing of the products output by additive manufacturing, and it processes the shape of the parts through fumigation and polishing. The inspection module inspects the products through a three-sided vision machine, and the packing module performs the packing task with a six-axis robot. There are two assembly modules (Types 1 and 2), and each shape of the part or sub-assembly has an appropriate assembly method. The general production process is to print parts from the additive manufacturing module and perform post-processing and assembly processes through the buffer. Post-processing and assembly are conducted sequentially according to the predefined process; then, the products are inspected and packed before being shipped.

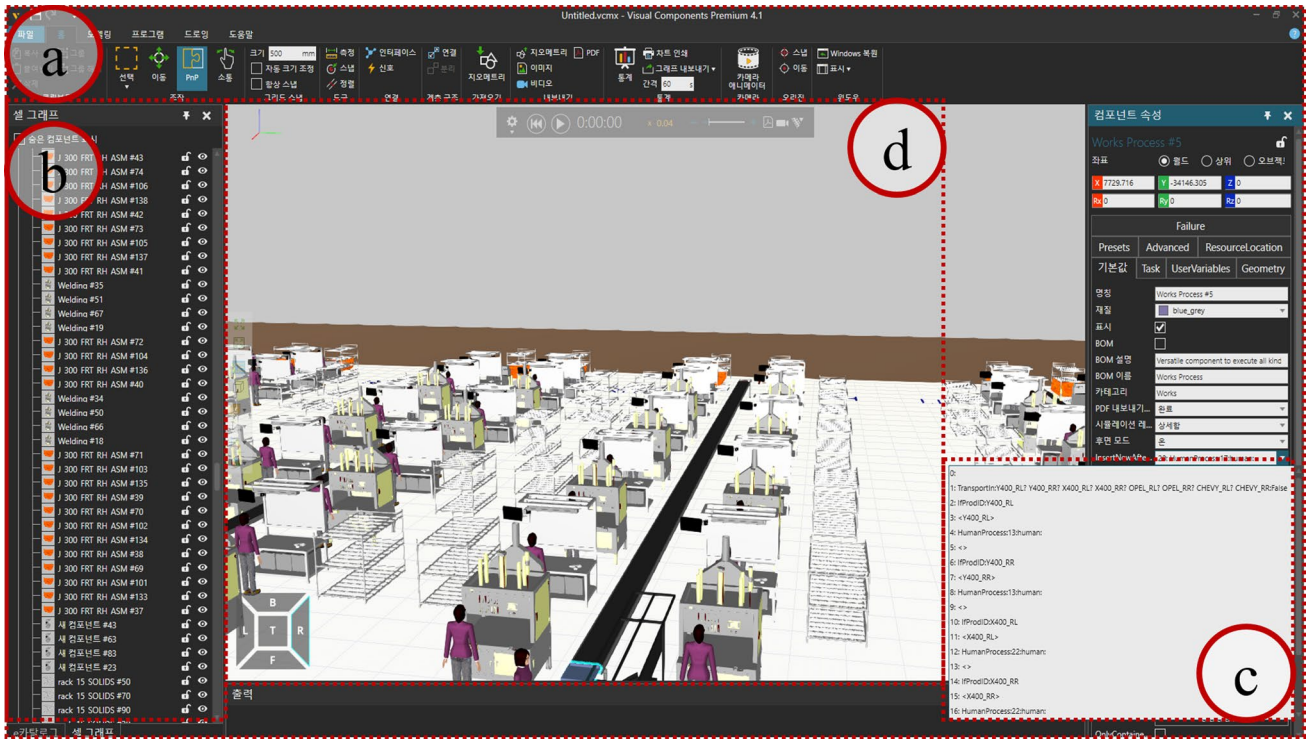

Fig. 8 Implemented DT application; created and synchronized virtual factory in the vehicle component manufacturing system

Table 10 Comparison between asset descriptions for the vehicle component manufacturing system (unit: %)

Item	VREDI	NESIS	CMSD
Number of information objects	100.00	172.69	131.33
Unique information objects utilized	100.00	89.08	102.48
Unique information objects without duplication	100.00	51.58	78.03

The main problem for the productivity of the MSF occurs in the post-processing station. The additive manufacturing module has the longest processing time, but has sufficient production capacity because of the installation of twelve resources. Therefore, the layout and logistics around post-processing station should be designed to achieve the maximum efficiency within the constraints of cost and space. Furthermore, because it is an unmanned work center, there is no resident worker. In addition, as it uses additive manufacturing as the main production method, the defect rate is not low and often causes a setback in the overall production plan. Further, it has difficulties in scheduling and responses because of product diversity at the personalization level. Thus, it is difficult to cope immediately with an abnormality, and the installation of cameras for remote monitoring cannot provide diverse information around the operations in the MSF.

Table 11 shows the types of the products produced in the MSF and their process plans. The work center handles various types of parts even if it produces small products, as it produces a variety of personalized products. The products in Table 11 are those used for the analysis of the results of this industrial case study.

The material flow in the MSF for the parts in Table 11 is summarized in a from/to chart, presented in Table 12. This shows the material handling operations that occur frequently in the MSF. Each part is first processed in the additive manufacturing module, followed by the buffer. The number of parts that move from the buffer to the polishing module was the largest, accounting for 27.714% of the total, and they showed even movements. The parts moving from the polishing module to assembly module no. 2 accounted for 12.672%, and the parts moving from the fumigation module to assembly module no. 1 accounted for 19.506%, which represented the major movements of parts. Therefore, the current MSF layout has many inefficient material flow robot movements because of the distance between the frequently moving modules.

Application information

Table 13 summarizes the detailed application information for vertical integration and horizontal coordination. Interoperability is achieved using web communication to overcome

Fig. 9 MSF in Daejeon, Republic of Korea

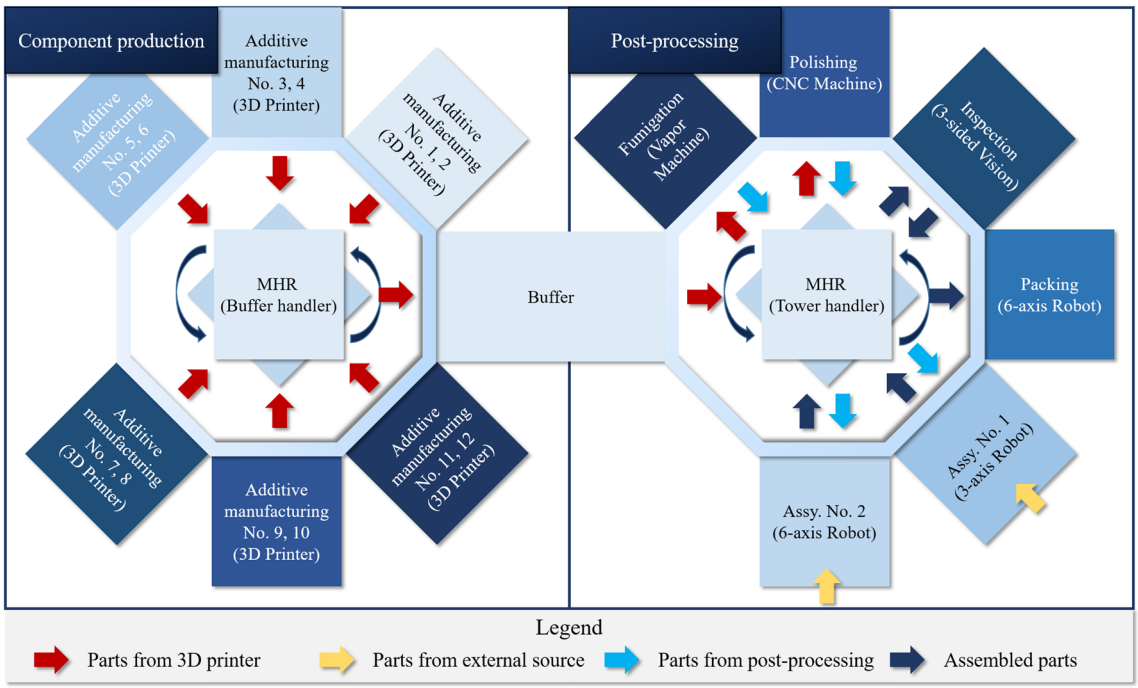

Fig. 10 Layout of the MSF (Park et al. 2019b, 2020; Kang et al. 2018)

the heterogeneous development environments and characteristics. The IIoT network was designed to aggregate manufacturing data using a RESTful API, and the IIoT devices of each system were composed using embedded firmware, Socket API, and DI/O module. The detailed descriptions of the application information of the component manager and DT application are as follows:

- *Component manager and service hosts* These components were implemented using a WCF framework. Because each application had its own heterogeneous development environments, languages, and core functional engines, the WCF framework was selected for enabling loosely-coupled integration based on a web service. The manifest, binding, and contract information

Table 11 Product and process plan information of the MSF

Product	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7	No. 8	Rate (%)
1	Additive	Polishing	Assembly No. 1	Inspection	Packing	–	–	–	4.980
2	Additive	Polishing	Assembly no. 2	Inspection	Packing	–	–	–	5.975
3	Additive	Polishing	Assembly no. 1	Polishing	Inspection	Packing	–	–	1.696
4	Additive	Polishing	Assembly no. 1	Assembly no. 2	Inspection	Packing	–	–	0.035
5	Additive	Polishing	Assembly no. 1	Fumigation	Inspection	Packing	–	–	2.186
6	Additive	Polishing	Assembly no. 2	Polishing	Inspection	Packing	–	–	3.132
7	Additive	Polishing	Assembly no. 2	Fumigation	Inspection	Packing	–	–	1.969
8	Additive	Polishing	Fumigation	Assembly no. 1	Inspection	Packing	–	–	3.783
9	Additive	Polishing	Fumigation	Assembly no. 2	Inspection	Packing	–	–	1.226
10	Additive	Polishing	Assembly no. 1	Fumigation	Assembly no. 2	Inspection	Packing	–	0.091
11	Additive	Polishing	Fumigation	Assembly no. 1	Assembly no. 2	Inspection	Packing	–	0.260
12	Additive	Polishing	Fumigation	Assembly no. 2	Assembly no. 1	Inspection	Packing	–	1.448
13	Additive	Polishing	Assembly no. 2	Polishing	Assembly no. 1	Fumigation	Inspection	Packing	0.241
14	Additive	Polishing	Assembly no. 2	Polishing	Assembly no. 2	Polishing	Inspection	Packing	0.677
15	Additive	Assembly no. 1	Inspection	Packing	–	–	–	–	16.647
16	Additive	Assembly no. 1	Polishing	Inspection	Packing	–	–	–	1.696
17	Additive	Assembly no. 1	Assembly no. 2	Inspection	Packing	–	–	–	0.801
18	Additive	Assembly no. 1	Fumigation	Inspection	Packing	–	–	–	4.715
19	Additive	Assembly no. 1	Fumigation	Assembly no. 2	Inspection	Packing	–	–	0.091
20	Additive	Assembly no. 2	Inspection	Packing	–	–	–	–	25.437
21	Additive	Assembly no. 2	Assembly 1	Inspection	Packing	–	–	–	3.425
22	Additive	Assembly no. 2	Polishing	Inspection	Packing	–	–	–	0.964
23	Additive	Assembly no. 2	Fumigation	Inspection	Packing	–	–	–	2.034
24	Additive	Assembly no. 2	Polishing	Assembly no. 1	Fumigation	Inspection	Packing	–	0.241
25	Additive	Fumigation	Assembly no. 1	Inspection	Packing	–	–	–	15.424
26	Additive	Fumigation	Assembly no. 2	Inspection	Packing	–	–	–	0.785
27	Additive	Fumigation	Assembly no. 1	Assembly no. 2	Inspection	Packing	–	–	0.039

Table 12 Material flow summary of the MSF (unit: %)

From	To						
	Buffer	Polishing	Fumigation	Assembly no. 1	Assembly no. 2	Inspection	Packing
Buffer	–	27.741	16.249	23.95	32.102	–	–
Polishing	–	–	6.717	9.471	12.672	8.165	–
Fumigation	–	–	–	19.506	3.643	11.386	–
Assembly no. 1	–	3.391	7.325	–	1.135	41.925	–
Assembly no. 2	–	5.934	4.003	4.873	–	34.741	–
Inspection	–	–	–	–	–	–	100.000
Packing	–	–	–	–	–	–	–

was prepared in XML format for universal use and practical application.

- *DT application* This application was implemented based on the proposed architectural framework. The DT engine of the DT application was selected to apply DMWorks for MSF. (1) This DT engine is a robotics simulation engine and is suitable for MSF, which executes the robotics operations of MHRs. (2) In addition, the DT engine provides a sufficient API library in the C++ assembly language for efficient development. (3) Moreover, admirable 3D visualization of this DT engine can support the monitoring functionality efficiently. The operation module of the DT application was implemented in C++ assembly language with the MFC model as those are compatible with the API library of the DT engine. The base models in CDL are implemented to enable universal use according to the STEP standard. Additionally, the logic and metadata in CDL are implemented in XML for-

Table 13 Application information of components in the MSF

Component	Item	Contents
IIoT network	Network architecture	RESTful API
IIoT device	Buffer handler robot	Embedded firmware
	Tower handler robot	Socket API
	Additive manufacturing	Serial communication
	Fumigation	DI/O module
	Polishing	DI/O module
	Inspection	Socket API
	Packing	Embedded firmware
	Assembly no. 1	Socket API
	Assembly no. 2	Embedded firmware
DT application	Development environment	Visual Studio 2015
	Programming language	C++
	Programming model	MFC
	DT engine	ezRobotics DMWorks × 64 2.4.15826.0
	Base model	STEP
	Meta data	XML
	Logic	XML
Resource allocation application	Development environment	Visual Studio 2015
	Programming language	C#
	Programming framework	.NET Framework 4.6.2
	Core functional engine	Simulation-based integer programming
Factory logistics design application	Development environment	Visual Studio 2015
	Programming language	C#
	Programming framework	.NET Framework 4.6.2
	Core functional engine	Simulation-based genetic algorithm
Planning and scheduling application	Development environment	Visual Studio 2015
	Programming language	C#
	Programming framework	.NET Framework 4.6.2
	Core functional engine	Simulation-based genetic algorithm
Diagnosis application	Development environment	Visual Studio 2015
	Programming language	C++
	Core functional engine	RBR
Dynamic response application	Development environment	Visual Studio 2015
	Programming language	C#
	Programming framework	.NET Framework 4.6.2
	Core functional engine	Simulation-based genetic algorithm
Time-machine monitoring application	Development environment	Visual Studio 2015
	Programming language	C++
	Core functional engine	RBR
Component manager	Development environment	Visual Studio 2015
	Programming language	C#
	Programming model	WCF
	Programming framework	.NET Framework 4.6.2
	Manifest	XML
	Binding	XML
	Contract	XML

Fig. 11 Base models for the DT application of the MSF (Park et al. 2019b, 2020)

mat for universal use. Figure 11 shows these base models of the DT application. All base models of the modules of the two stations were constructed. These base models are imported when the operation module creates a DT based on the metadata.

Among the core functional engines of each application, the information about the core functional engine of the work center design stage, which is the type stage, is as follows.

- *Resource allocation application* This application was implemented to determine the machine number in order to find the number of resources required to achieve robustness of various products. The number of machines was found only for the post-processing station, and indicated the number of machines that could enter the module, and not the total number of modules. In other words, the objective of core functional engine of this application was to maximize the average operation rate for the multiple cases in Table 11. The objective is derived by the simulation based on the implemented DT. This application is coordinated with VREDI by changing the number of 'Machine' elements of *Plant* class. Production plan cases with severe loads were input into each module. The constraints on the machines at the post-processing station are as follows: (1) the fumigation and polishing module can be configured as double-layered modules, and (2) the other machines are configured as single-layered modules. In addition, (3) the total number of machines in post-processing station is limited to less than eight. Furthermore, (4) one or more modules must be included for each type of module in the post-processing station.
- *Factory logistics design application* This application aimed to minimize the total transportation distance of an MHR based on limitation of the number of modules at the post-processing station, and the product and process plan information for a specific period, which is described in Tables 11 and 12. Unlike resource allocation, the number of modules was set to be the variable in the algorithm instead of the number of resources. The

objective of this core functional engine was to minimize the transportation distance of the tower handler robot, for manufacturing the products according to Table 11. The objective is also provided by the simulation based on the implemented DT. This application is coordinated with VREDI by changing the location and transformation information of 'Machine' element of *Plant* class. The length of the chromosome is the number of resources in the post-processing station, and each gene indicates the location of a module in the post-process station.

Figure 12 shows the procedure for the service composition of applications, that is, horizontal coordination for DT-based technical functionalities in the type stage. First, the resource allocation application determines the number of machines in post-process station in the MSF. The initial solution of the factory logistics design is created by using the number of machines determined by the resource allocation application. Based on the technical functionalities of two advanced engineering applications, the P4R information object is configured for the DT application. The DT is created and synchronized in the DT application according to the P4R information object reflecting the technical functionalities. The DT returns simulation result that contains objective of the two applications by DT utilization procedures. Once the optimal factory logistics are designed, the final DT is created and synchronized based on these logistics.

Among the core functional engines of each application, the information of the core functional engine of the work center operation stage, which is the instance stage, is as follows.

- *Planning and scheduling application* This application aimed to lower the load factor of each process operation for planning, and lower the total makespan for scheduling. To that end, planning was established to minimize the variance of the load factors of each process module of the manufactured products within the range of the production plan. Additionally, scheduling was designed to derive the production sequence using NSGA II (Deb et al. 2000) based on the production plan by date. The makespan, which is the objective of scheduling of this application, is derived by the simulation result of the DT application. This application is coordinated with VREDI by changing the information of 'TargetVolume' element of *Plan* class and input operation information of 'SourceInputInfo' element of *Process* class.
- *Time-machine notification application* The RBR rules were designed to re-aggregate the information object of history, aggregate information object of the present production operation, and synchronize the future production plan to the DT. The past and current information were identified based on the 'RegisteredTime' of the Header of VREDI, and the DT was created and synchronized on the basis of the registered elements of the corresponding time point.
- *Diagnosis application* For this application, the RBR rule was specified to reflect and identify the possibility of normal performance of the plan in the MSF according to the change in the elements in VREDI. When there was a change in the value of any element in the 'WorkCalendar,' 'Procurement,' and 'TargetVolume' elements of the *Plan* class, it was identified as an event in the production plan. When 'FailureStatus' of the *Plant* class was identified as a failure, it was set as a machine failure event, and when 'PoorQuality' was identified, it was set as a quality problem event. When 'ProceedTime' and 'StockNum' in the *Plant* class were delayed, they were set as a delay event of the production plan. As mentioned above, the 'diagnosis service' of the work-center-level checks the 'TargetVolume' element in the *Plan* class, and compares this element to the simulation results to verify whether this plan can be continued feasibly or it is needed to establish a reactive plan according to the event identification result.
- *Dynamic response application* This application executes a procedure for creating a reactive production plan within the given range when the 'diagnosis service' determines that it needs a change in plan and schedule for the current production operation. This application executes a procedure for recreating the plan and schedule that were created by the planning and scheduling application and the DT application, because it is a discrete manufacturing system.

Figure 13 shows the horizontal coordination procedure of the work center operation stage, which is the instance stage. This work center operation stage can be divided into the operation planning phase and the operation execution phase. In the operation planning phase, the production plan by date and the initial sequence are derived from the planning and scheduling applications. The production plan is drafted by creation, synchronization, and utilization of the work-center-level DT. In the operation execution phase, VREDI is re-aggregated continuously in accordance with the predefined time points. The 'time-machine monitoring' service for various selected time points is visualized for users by synchronizing the re-aggregated information object to the DT. Then, the situation in the work center is diagnosed. This re-aggregated VREDI and the diagnosis results are synchronized to the DT and visualized to users. If a defined abnormal situation is diagnosed, the DT application verifies its effect. If the production is impossible, it executes a procedure that creates an alternative production plan within the given range.

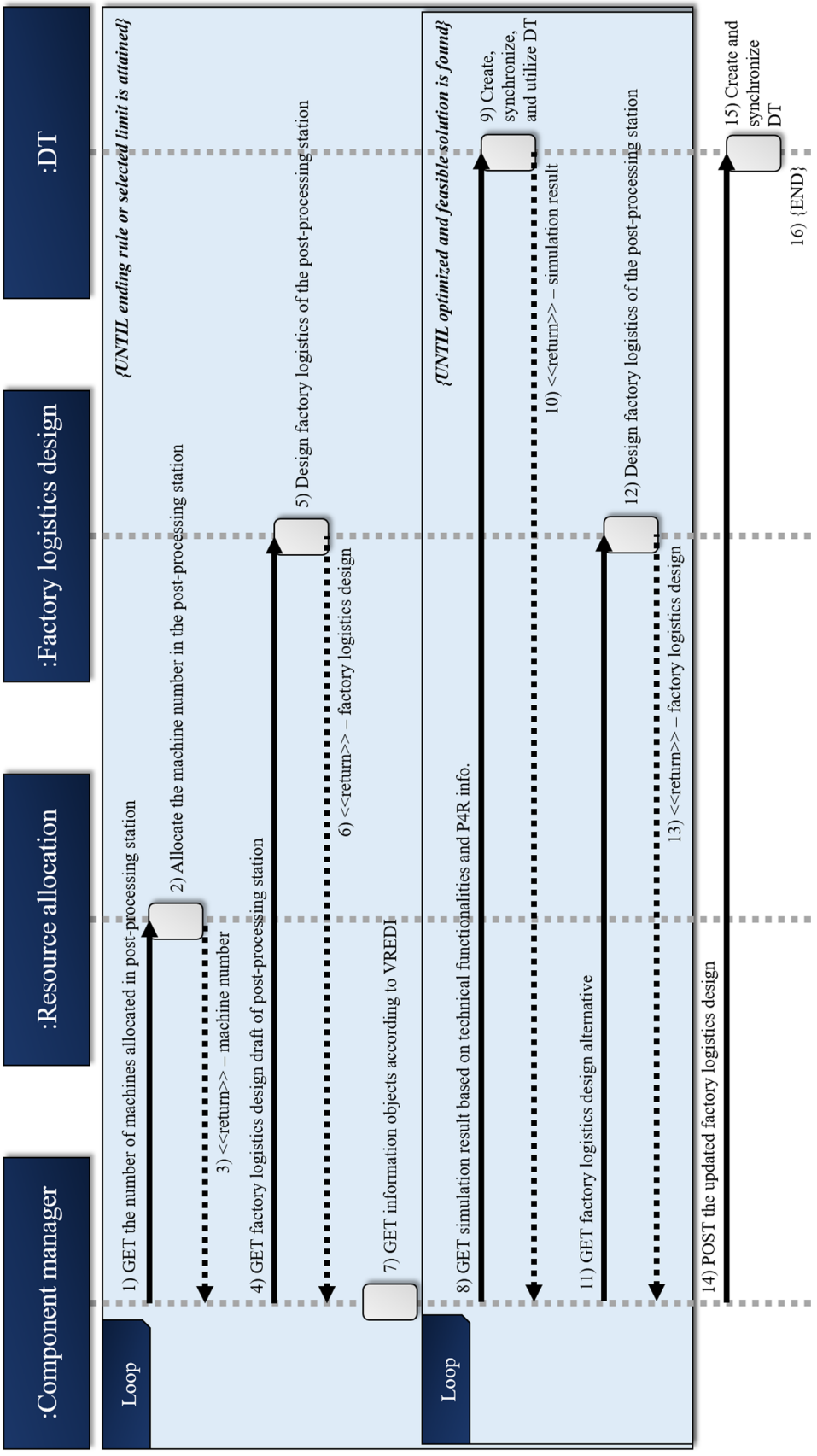

Fig. 12 Horizontal coordination procedure of the DT-based technical functionality in the type stage

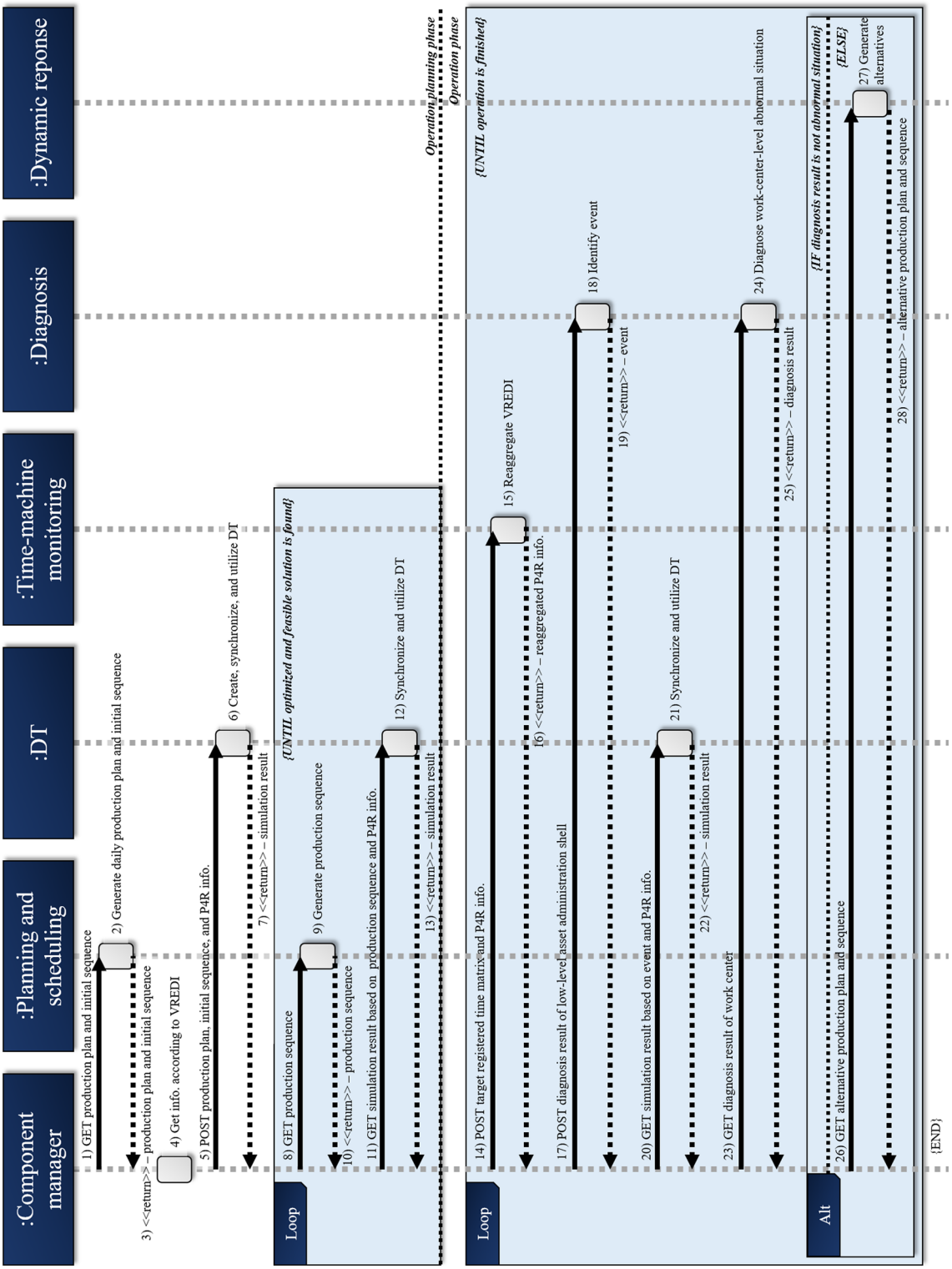

Fig. 13 Horizontal coordination procedure of DT-based technical functionality in the instance stage

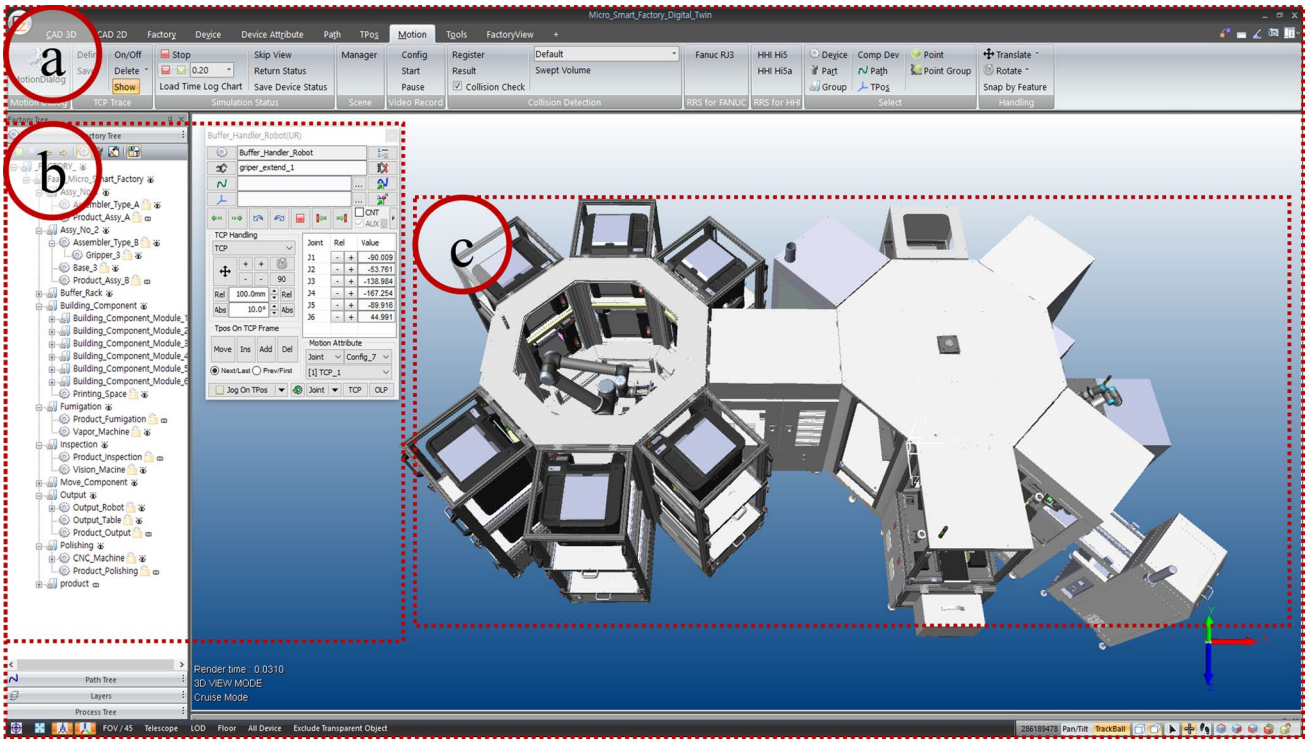

Fig. 14 Implemented DT application; created and synchronized DT in the MSF

Application for DT-based technical functionality

Figure 14 shows the actual DT application implemented, and the created and synchronized DT based on VREDI. Figure 14a shows DMWorks, the DT engine of the DT application; Fig. 14b shows the screen on which logic and metadata are imported; and Fig. 14c shows a screen where the DT of the MSF is created based on VREDI by importing the base model. As mentioned above in the vehicle component manufacturing system, this shows that VREDI normally supports the creation and synchronization procedures of the DT application.

Figure 15 shows the result of the horizontal coordination procedure of the work center design stage, which is the type stage shown in Fig. 11. The 'resource allocation service' and 'factory logistics design service' were executed, and the existing work center was improved, as shown by the text in the white boxes. The 'resource allocation service' added one vapor machine and one CNC machine in the MSF. In addition, the 'factory logistics design service' placed modules that moved materials frequently close to each other to increase the material flow efficiency. The simulation result of production operation for the target products in Table 11 was 9877 min compared to 11,064 min for the makespan of the existing layout of the MSF. Thus, the makespan showed an improvement of approximately 12.018%.

Figure 16 shows a screen on which the horizontal coordination procedure of the operation planning phase of the work center operation stage is executed. The virtual factory is not used by the 'resource allocation service' and 'factory logistics design service' because of the interface with the field. The production plan by date and the input sequence provided by the planning and scheduling application are input into the DT, as shown in Fig. 16a. When the simulation is completed, the simulation results are output, as shown in Fig. 16b. When an unfeasible production plan or input sequence is entered, an error message is displayed, as shown in Fig. 16c.

Figure 17 shows the screen for running the horizontal coordination procedure of the operation execution phase in the work center. Figure 17a shows the visualization of the current operation status of the DT synchronized to the actual field through VREDI; Fig. 17b, c show that the 'time-machine monitoring service' is provided for the existing production plan and current operation situation; and Fig. 17d shows that the current event and the diagnosis results for the possibility of normal performance of the production plan through the 'diagnosis service' is synchronized and visualized to users. Figure 17c shows the existing production plan and a difference from the current operation status in Fig. 17b, which indicates that it was caused by a system failure, as reported in Fig. 17d. Subsequently, the dynamic response application provides an alternative solution by

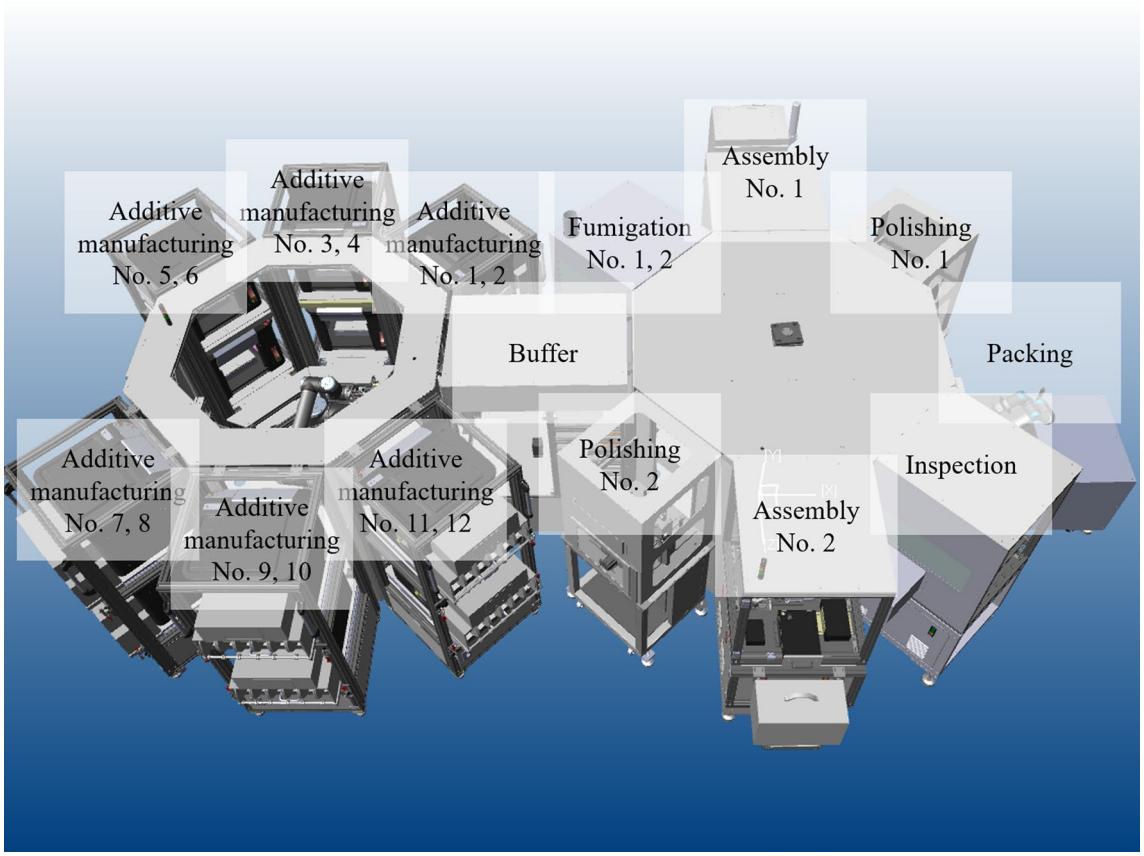

Fig. 15 The updated MSF layout based on resource allocation and factory logistics design service

performing the horizontal coordination of the planning and scheduling application and the DT application. These dynamic services in the operation execution phase exhibit a higher efficiency than the static production operation status of an existing work center. When dynamic responses are included in the schedule created in the 'planning and scheduling service,' the makespan improves by 6.164% from 10,988 min to 10,350 min. The main reason for the improvement evaluated in this experiment was that the fluctuation in production due to the high defect rate of the 3D printers was improved by applying these DT-based technical functionalities. One of the reasons for this is that the high capacity of the component production station was not used properly for static responses.

Analysis of the results of industrial case studies

The industrial case studies were conducted in two work centers to ensure that all four core technical requirements of VREDI were met: (a) DT definition, (b) Inheritance of AAS property, (c) Improvement in existing asset descriptions, and d) DT-based technical functionality. The case studies showed the different DT applications implemented based on the proposed architectural framework in a heterogeneous application and development environment. Moreover, the industrial case studies also validated the two requirements of DT definition and improvement over existing asset descriptions. The first industrial case study was performed to validate the inheritance of the AAS property, and the second industrial case study was conducted to validate DT-based technical functionality. The results of the case studies confirmed the following:

(a) The proposed VREDI was confirmed to meet the definition of DT: automatic creation, vertical integration support, horizontal coordination support, and representation of an information object for the core technical functionality of the DT in order to derive the performance indicator. (1) The proposed VREDI effectively supports the creation and synchronization of DTs in both case studies. (2) Vertical integration and horizontal coordination support is confirmed through the designed service composition based on the component manager. (3) Using virtual representation objects based on VREDI, the DT application can provide a simulation-related technical functionality for providing

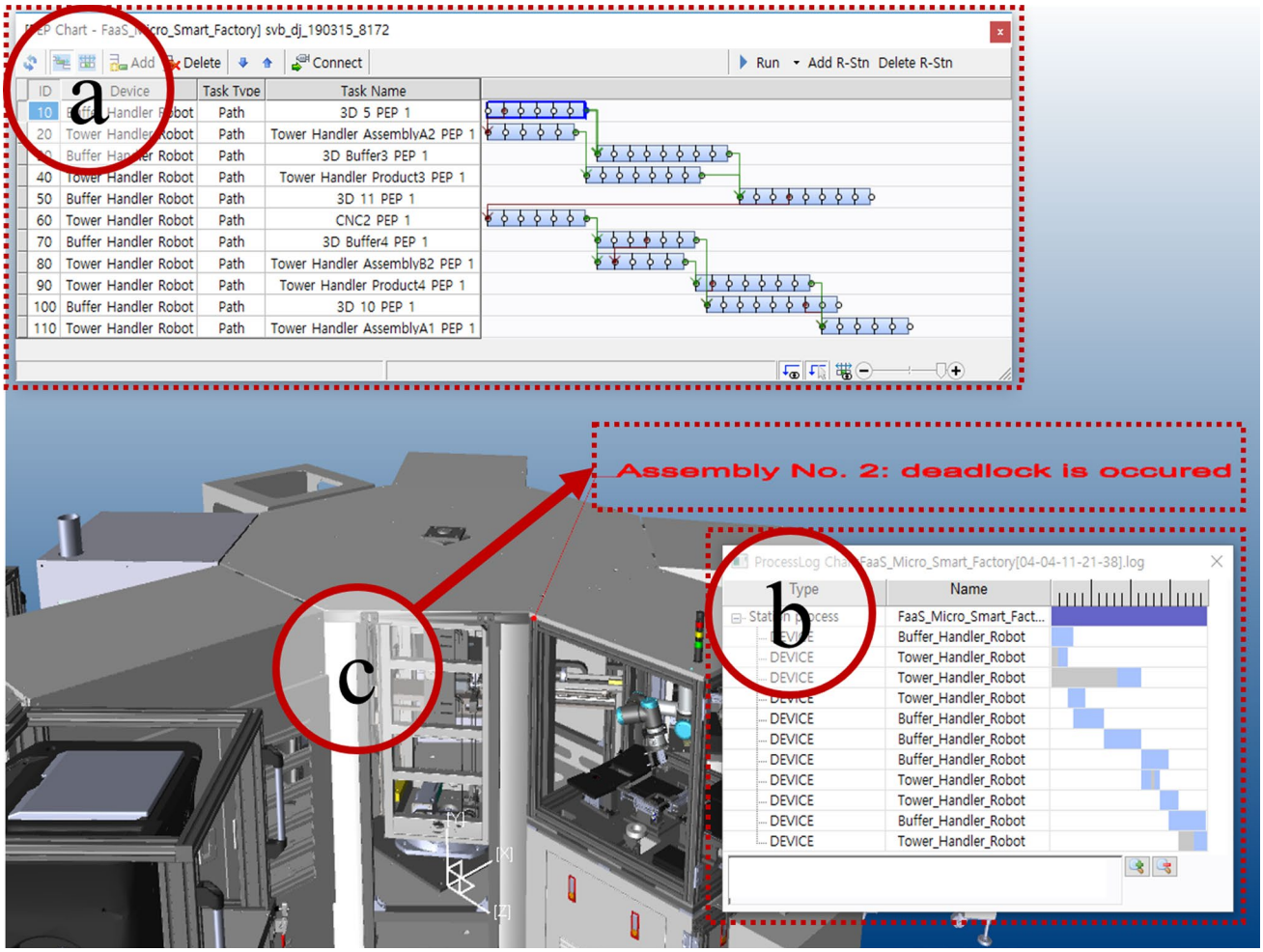

Fig. 16 Planning and scheduling service based on VREDI

contract information from the component manager to enable service composition.

(b) By inheriting the AAS property to the VREDI, the proposed virtual representation object and DT application are effectively implemented and applied in two work centers. Additionally, the results confirmed that VREDI can support reasonable interoperation with various engineering applications. Moreover, the component manager can contract and request information from virtual representations in heterogeneous levels of AAS and technical functionality from engineering applications.

(c) In both the industrial case studies, the virtual representation object with suitable P4R elements for each work center was aggregated and instantiated. The virtual representation objects were implemented based on the XML format, which is suitable for the SOAP protocol. By implementing VREDI with detailed and clear definitions of each element, the proposed asset description could effectively describe the virtual factory of the component manufacturing system with fewer duplicated information objects than NESIS and CMSD. In comparison with VREDI, NESIS and CMSD modeled the virtual factory of the vehicle component manufacturing system with approximately 48.42% and 21.97% more duplicated information objects, respectively. In addition, the ratio of information managed by the work-center-level AAS was approximately 62.49%, which confirms the effectiveness of component-enabled aggregation and the "type and instance" concept. Thus, it was found that VREDI can ensure reasonable usage of information in information objects to be input to a DT application. The description of VREDI was more specialized to the DT than that of NESIS and CMSD; moreover, VREDI had additional elements such as an energy-related property and a material plan. It also contained elements for synchronization of field information, which could not be realized for other asset descriptions.

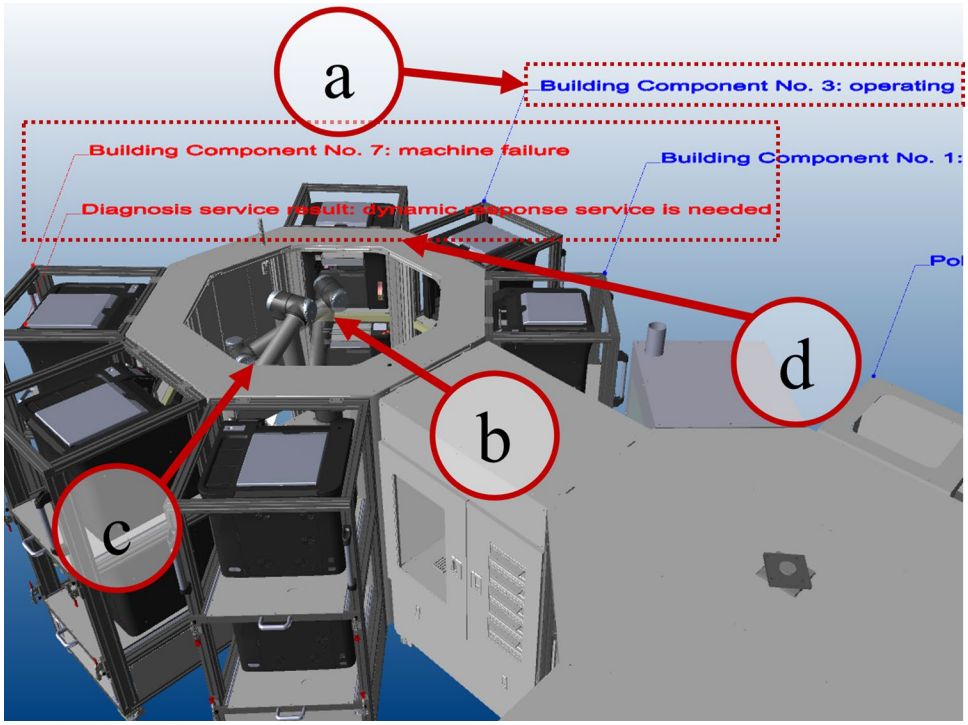

Fig. 17 Time-machine monitoring, diagnosis, and dynamic response service based on VREDI

(d) Six DT-based technical functionalities (i.e., technical functionalities in the work-center-level AAS selected from various DT-based technical functionalities) were applied to an MSF as an industrial case study. When the products defined in Table 11 were produced, the technical functionalities of the type stage showed an improvement in the makespan by approximately 12.018%, and the technical functionalities of the instance stage showed an improvement in the makespan by approximately 6.164%. These results demonstrate the effectiveness of the six DT-based technical functionalities applied to the MSF.

Conclusion

This paper proposed VREDI, which is an asset description for a DT application in the work-center-level AAS of RAMI 4.0. To rationalize the design of the VREDI, four core technical requirements were analyzed: (a) DT definition, (b) Inheritance of AAS property, (c) Improving the existing asset description, and d) DT-based technical functionality. In addition, six DT-based technical functionalities were proposed to improve the efficiency of the work center design stage (the type stage), and the work center operation stage (the instance stage). The architectural framework for implementing the DT-based technical functionalities and the interoperations of the core functional engine of the advanced engineering application coordinated with the DT application were proposed. The object-oriented P4R classes were applied with the "type and instance" concepts of VREDI. Moreover, the relationships and detailed attribute descriptions among these classes were explained. Furthermore, the elements in each class and the level of AAS to which these elements belong were described through a manifest. VREDI was accordingly applied to two work centers in order to confirm that it meets the four core technical requirements.

To support the information usage of DT as a smart factory technology, the abovementioned various aspects of its requirements were analyzed. VREDI was applied with concepts to support the main properties of DT, and was designed to inherit the properties of AAS for achieving efficient information management and interoperability. The application of advanced concepts overcame the limitations of existing asset descriptions, and designated elements to support technical functionality. The detailed contributions of this study are as follows:

I. The VREDI is designed to provide an information object to a DT application that accommodates the four properties of the DT: automatic creation, supporting vertical integration, supporting horizontal coordination, and indicating and representing an information object for the core technical functionality of DT in order to derive the performance indicator.
II. By inheritance of the AAS properties, VREDI can utilize component-manager-enabled aggregation, which allows managing and using distributed infor-

mation objects. For other improvements resulting from applying the AAS concept, VREDI follows SOA principles for high-level interoperability, and applies the "type and instance" concept to reduce the usage of duplicated information objects.

III. VREDI provides improved asset description compared to other existing descriptions, as it contains information elements related by vertical integration and horizontal coordination; applies the "type and instance" concept; considers DT-based technical functionality; and presents all major elements.

IV. Six DT-based technical functionalities were suggested and divided into type and instance stages in the work-center-level value stream, which provides a reference case for other studies. The technical functionalities are considered to derive core elements of P4R structure and technical requirements. In addition, the proposed VREDI contains elements for representation of service composition to satisfy a rational service-oriented design.

V. An early case of the detailed work-center-level AAS application was discussed. Additionally, it is expected to provide a reference case for other studies because of the detailed design and application.

This study confirmed that the AAS model sufficiently supports the implementation of smart factory technologies by several advanced concepts, such as SOA, asset description, and loosely-coupled integration of heterogeneous elements. The core features of VREDI, such as component-manager-enabled aggregation, were enabled by the core features of the AAS. It is believed that the core features of the AAS should be employed to achieve the smooth convergence of technologies applied to smart factories as well as the DT. However, certain parts of the AAS require clearer definitions. The technical functionality is defined as a physical application in some papers, and it is specified based on the service or function provided by the components. In this study, it was specified according to the latter, but this requires a clear definition to advance from a reference model to concrete standards.

In further research, the expansion of VREDI to the supply-chain-level is required to support distributed DT simulation. This research direction can determine an effective scenario for supporting high-level interoperability. In addition, research should also be pursued to develop an information-fusion method that allows VREDI to obtain useful information from raw data, such as the information obtained by standards-based IoT or legacy systems. This method would help to improve the ease of applying the proposed VREDI and self-configuration of DT in DT application. Furthermore, a VREDI should be specified for a reconfigurable manufacturing system in which the two services, presented as technical functionalities in the type stage in this study, are used in the instance stage.

Acknowledgements This work was supported by the IT R&D Program of MOTIE/KEIT (10052972, Development of the Reconfigurable Manufacturing Core Technology Based on the Flexible Assembly and ICT Converged Smart Systems) and the WC300 Project (S2482274, Development of Multi-vehicle Flexible Manufacturing Platform Technology for Future Smart Automotive Body Production) funded by the Ministry of SMEs and Startups.

References

Adolphs, P., Auer, S., Bedenbender, H., Billmann, M., Hankel, M., Heidel, R., et al. (2016). Structure of the administration shell continuation of the development of the reference model for the Industrie 4.0 component. ZVEI and VDI, status report.

Alam, K. M., & El Saddik, A. (2017). C2PS: A digital twin architecture reference model for the cloud-based cyber-physical systems. *IEEE Access, 5,* 2050–2062.

Bandyopadhyay, D., & Sen, J. (2011). Internet of things: Applications and challenges in technology and standardization. *Wireless Personal Communications, 58*(1), 49–69.

Bedenbender, H., Billmann, M., Epple, U., Hadlich, T., Hankel, M., Heidel, R., et al. (2017). Examples of the asset administration shell for Industrie 4.0 components—Basic part. ZVEI white paper.

Bloomfield, R., Mazhari, E., Hawkins, J., & Son, Y. J. (2012). Interoperability of manufacturing applications using the core manufacturing simulation data (CMSD) standard information model. *Computers and Industrial Engineering, 62*(4), 1065–1079.

Cadavid, J. P. U., Lamouri, S., Grabot, B., Pellerin, R., & Fortin, A. (2020). Machine learning applied in production planning and control: A state-of-the-art in the era of industry 4.0. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-019-01531-7.

Cheng, Y., Zhang, Y., Ji, P., Xu, W., Zhou, Z., & Tao, F. (2018). Cyber-physical integration for moving digital factories forward towards smart manufacturing: A survey. *The International Journal of Advanced Manufacturing Technology, 97*(1–4), 1209–1221.

Chuang, A. C. C. (2016). Discuss the standard of Industry 4.0. Hua University, Taiwan. http://www.ebc.nthu.edu.tw/StudentProject/eifinal/2015_eiProject/word/07.pdf. Retrieved May 9, 2019.

da Cruz, M. A., Rodrigues, J. J. P., Al-Muhtadi, J., Korotaev, V. V., & de Albuquerque, V. H. C. (2018). A reference model for internet of things middleware. *IEEE Internet of Things Journal, 5*(2), 871–883.

Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International conference on parallel problem solving from nature* (pp. 849–858). Berlin: Springer.

Derigent, W., Cardin, O., & Trentesaux, D. (2020). Industry 4.0: Contributions of holonic manufacturing control architectures and future challenges. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-020-01532-x.

Dorst, W. (Ed.). (2015). *Umsetzungsstrategie Industrie 4.0: Ergebnisbericht der Plattform Industrie 4.0*. Berlin: Bitkom Research GmbH.

Fleischmann, H., Kohl, J., & Franke, J. (2016). A reference architecture for the development of socio-cyber-physical condition monitoring systems. In *2016 11th system of systems engineering conference (SoSE)* (pp. 1–6). IEEE.

Gabor, T., Belzner, L., Kiermeier, M., Beck, M. T., & Neitz, A. (2016). A simulation-based architecture for smart cyber-physical systems.

In *2016 IEEE international conference on autonomic computing (ICAC)* (pp. 374–379). IEEE.
Grieves, M. (2014). Digital twin: Manufacturing excellence through virtual factory replication. *White Paper, 1,* 1–7.
Hankel, M., & Rexroth, B. (2015). The reference architectural model industrie 4.0 (rami 4.0). ZVEI, April.
IEC. (2016). IEC TS 62832-1:2016 industrial-process measurement, control and automation—Digital factory framework—Part 1: General principles. https://webstore.iec.ch/publication/33023. Retrieved May 2, 2019.
Jeon, B., Yoon, J. S., Um, J., & Suh, S. H. (2020). The architecture development of Industry 40 compliant smart machine tool system (SMTS). *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-020-01539-4.
Kagermann, H., Helbig, J., Hellinger, A., & Wahlster, W. (2013). *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry; final report of the Industrie 4.0 Working Group*. Forschungsunion.
Kang, H. S., Noh, S. D., Son, J. Y., Kim, H., Park, J. H., & Lee, J. Y. (2018). The FaaS system using additive manufacturing for personalized production. *Rapid Prototyping Journal, 24*(9), 1486–1499.
Kim, D. Y., Park, J. W., Baek, S., Park, K. B., Kim, H. R., Park, J. I., et al. (2020). A modular factory testbed for the rapid reconfiguration of manufacturing systems. *Journal of Intelligent Manufacturing, 31*(3), 661–680.
Komoda, N. (2006). Service oriented architecture (SOA) in industrial systems. In *2006 4th IEEE international conference on industrial informatics* (pp. 1–5). IEEE.
Lee, J. Y., Kang, H. S., Kim, G. Y., & Do Noh, S. (2012). Concurrent material flow analysis by P3R-driven modeling and simulation in PLM. *Computers in Industry, 63*(5), 513–527.
Lee, J. Y., Kang, H. S., Noh, S. D., Woo, J. H., & Lee, P. (2011). NESIS: A neutral schema for a web-based simulation model exchange service across heterogeneous simulation software. *International Journal of Computer Integrated Manufacturing, 24*(10), 948–969.
Li, L. (2018). China's manufacturing locus in 2025: With a comparison of "Made-in-China 2025" and "Industry 4.0". *Technological Forecasting and Social Change, 135,* 66–74.
Liao, Y., Loures, E. D. F. R., & Deschamps, F. (2018). Industrial internet of things: A systematic literature review and insights. *IEEE Internet of Things Journal, 5*(6), 4515–4525.
Lim, K. Y. H., Zheng, P., & Chen, C. H. (2019). A state-of-the-art survey of Digital Twin: techniques, engineering product lifecycle management and business innovation perspectives. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-019-01512-w.
Liu, Q., Zhang, H., Leng, J., & Chen, X. (2019). Digital twin-driven rapid individualised designing of automated flow-shop manufacturing system. *International Journal of Production Research, 57*(12), 3903–3919.
MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., Metz, R., & Hamilton, B. A. (2006). Reference model for service oriented architecture 1.0. *OASIS Standard*, *12*(S 18).
Mónica, R. L., Christian, B., Friedrich, M., Bernd, K., Urlich, B., & Waldemar, S. (2016). Agent-based communication to map and exchange shop floor data between MES and material flow simulation based on the open standard CMSD. *IFAC-PapersOnLine, 49*(12), 1526–1531.
Ngai, E. W. T., Moon, K. K., Riggins, F. J., & Candace, Y. Y. (2008). RFID research: An academic literature review (1995–2005) and future research directions. *International Journal of Production Economics, 112*(2), 510–520.
Nikolakis, N., Alexopoulos, K., Xanthakis, E., & Chryssolouris, G. (2019). The digital twin implementation for linking the virtual representation of human-based production tasks to their physical counterpart in the factory-floor. *International Journal of Computer Integrated Manufacturing, 32*(1), 1–12.
Oztemel, E., & Gursev, S. (2020). Literature review of Industry 4.0 and related technologies. *Journal of Intelligent Manufacturing, 31*(1), 127–182.
Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2007). Service-oriented computing: State of the art and research challenges. *Computer, 40*(11), 38–45.
Park, K. T., Im, S. J., Kang, Y. S., Noh, S. D., Kang, Y. T., & Yang, S. G. (2019a). Service-oriented platform for smart operation of dyeing and finishing industry. *International Journal of Computer Integrated Manufacturing, 32*(3), 307–326.
Park, K. T., Lee, J., Kim, H.-J., & Noh, S. D. (2020). Digital twin-based cyber physical production system architectural framework for personalized production. *The International Journal of Advanced Manufacturing Technology, 106*(5–6), 1787–1810.
Park, K. T., Nam, Y. W., Lee, H. S., Im, S. J., Noh, S. D., Son, J. Y., et al. (2019b). Design and implementation of a digital twin application for a connected micro smart factory. *International Journal of Computer Integrated Manufacturing, 32*(6), 596–614.
Perrey, R., & Lycett, M. (2003). Service-oriented architecture. In *Proceedings of the 2003 symposium on applications and the internet workshops, 2003* (pp. 116–119). IEEE.
Qi, Q., & Tao, F. (2018). Digital twin and big data towards smart manufacturing and industry 4.0: 360 degree comparison. *IEEE Access, 6,* 3585–3593.
Qi, Q., Tao, F., Zuo, Y., & Zhao, D. (2018). Digital twin service towards smart manufacturing. *Procedia Cirp, 72,* 237–242.
Ramollari, E., Dranidis, D., & Simons, A. J. (2007). A survey of service oriented development methodologies. In *The 2nd European young researchers workshop on service oriented computing* (Vol. 75).
Redelinghuys, A. J. H., Basson, A. H., & Kruger, K. (2019). A six-layer architecture for the digital twin: A manufacturing case study implementation. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-019-01516-6.
Riddick, F., & Lee, T. (2010). *Core manufacturing simulation data (CMSD): A standard representation for manufacturing simulation-related information*. Gaithersburg: National Institute of Standards and Technology.
Schleich, B., Anwer, N., Mathieu, L., & Wartzack, S. (2017). Shaping the digital twin for design and production engineering. *CIRP Annals, 66*(1), 141–144.
Schroeder, G. N., Steinmetz, C., Pereira, C. E., & Espindola, D. B. (2016). Digital twin data modeling with automationml and a communication methodology for data exchange. *IFAC-PapersOnline, 49*(30), 12–17.
SISO. (2010). SISO-STD-006-2010: Standard for commercial off-the-shelf (COTS) simulation package interoperability (CSPI) reference models. Simulation Interoperability Standards Organization.
SISO. (2012). SISO-STD-008-01-2012: Standard for core manufacturing simulation data—XML representation. Simulation Interoperability Standards Organization.
Son, J. Y., Kang, H. C., Bae, H. C., Lee, E. S., Han, H. N., Park, J. H., et al. (2015). IoT-based open manufacturing service platform for mass personalization. *The Journal of the Korean Institute of Communication Sciences, 33*(1), 42–47.
Suri, K., Cadavid, J., Alferez, M., Dhouib, S., & Tucci-Piergiovanni, S. (2017). Modeling business motivation and underlying processes for RAMI 4.0-aligned cyber-physical production systems. In *2017 22nd IEEE international conference on emerging technologies and factory automation (ETFA)* (pp. 1–6). IEEE.
Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H., & Sui, F. (2018). Digital twin-driven product design, manufacturing and service with big data. *The International Journal of Advanced Manufacturing Technology, 94*(9–12), 3563–3576.

Tao, F., & Zhang, M. (2017). Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing. *IEEE Access, 5,* 20418–20427.

Tao, F., Zhang, M., & Nee, A. Y. C. (2019). *Digital twin driven smart manufacturing*. Cambridge: Academic Press.

Thompson, K. D. (2014). *Smart manufacturing operations planning and control program*. Gaithersburg: National Institute of Standards and Technology.

Tong, X., Liu, Q., Pi, S., & Xiao, Y. (2019). Real-time machining data application and service based on IMT digital twin. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-019-01500-0.

Uhlemann, T. H. J., Schock, C., Lehmann, C., Freiberger, S., & Steinhilper, R. (2017). The digital twin: Demonstrating the potential of real time data acquisition in production systems. *Procedia Manufacturing, 9,* 113–120.

Vachálek, J., Bartalský, L., Rovný, O., Šišmišová, D., Morháč, M., & Lokšík, M. (2017). The digital twin of an industrial production line within the industry 4.0 concept. In *2017 21st international conference on process control (PC)* (pp. 258–262). IEEE.

Weber, C., Königsberger, J., Kassner, L., & Mitschang, B. (2017). M2DDM: A maturity model for data-driven manufacturing. *Procedia CIRP, 63,* 173–178.

Weyrich, M., & Ebert, C. (2015). Reference architectures for the internet of things. *IEEE Software, 33*(1), 112–116.

Wiktorsson, M., Noh, S. D., Bellgran, M., & Hanson, L. (2018). Smart factories: South Korean and Swedish examples on manufacturing settings. *Procedia Manufacturing, 25,* 471–478.

Xu, L. D., He, W., & Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics, 10*(4), 2233–2243.

Yaqoob, I., Ahmed, E., Hashem, I. A. T., Ahmed, A. I. A., Gani, A., Imran, M., et al. (2017). Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges. *IEEE Wireless Communications, 24*(3), 10–16.

Yoon, S., Um, J., Suh, S. H., Stroud, I., & Yoon, J. S. (2019). Smart Factory Information Service Bus (SIBUS) for manufacturing application: Requirement, architecture and implementation. *Journal of Intelligent Manufacturing, 30*(1), 363–382.

Zezulka, F., Marcon, P., Vesely, I., & Sajdl, O. (2016). Industry 40—An introduction in the phenomenon. *IFAC-PapersOnLine, 49*(25), 8–12.

Zhang, H., Liu, Q., Chen, X., Zhang, D., & Leng, J. (2017). A digital twin-based approach for designing and multi-objective optimization of hollow glass production line. *IEEE Access, 5,* 26901–26911.

Zhang, Y., Zhang, G., Wang, J., Sun, S., Si, S., & Yang, T. (2015). Real-time information capturing and integration framework of the internet of manufacturing things. *International Journal of Computer Integrated Manufacturing, 28*(8), 811–822.