Journal of Manufacturing Systems 60 (2021) 93–118

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

Construction method of shop-floor digital twin based on MBSE

Juan Liu, Jianhua Liu, Cunbo Zhuang *, Ziwen Liu, Tian Miao

Laboratory of Digital Manufacturing, School of Mechanical Engineering, Beijing Institute of Technology, Beijing, 100081, China

ARTICLE INFO

Keywords:
Digital twin
Shop-floor digital twin
Cyber-physical production system
Model-based systems engineering
System modeling language
MagicGrid

ABSTRACT

Digital twin (DT) technology is essential for achieving the fusion of virtual-real cyber-physical systems. Academics and companies have made great strides in the theoretical research and case studies of constructing the shop-floor digital twin (SDT), which is the premise of applying DT technology on the shop floor. A shop floor is a large complex system that involves many elements including people, machines, materials, methods, and the environment and processes, such as the technical flow, business process, logistics, and control flow. However, most of the developed cases lack a hierarchical, structured and modularized implementation framework for the development of an SDT system, which leads to problems such as a low reuse rate of the system blocks, lack of scalability, and high upgrade and maintenance costs. In response to these issues, we propose a construction method of the DT for the shop floor based on model-based systems engineering from the perspective of the system. In this method, a comprehensive DT model for the shop floor is gradually constructed by using system modeling language, the modeling method "MagicGrid," and the "V model" of systems engineering. The model includes four dimensions of the shop-floor requirements, structure, behavior, and parameters, as well as three stages (the problem domain, solution domain, and implementation domain), and connects nine steps of the "V model," including the system requirements, system architecture, subsystem implementation, subsystem integration, and system verification. Then, based on an example of a real NC machining shop floor, subsystems including a visualization system, synchronization system, and simulation system, are discussed. Finally, the functions of the integrated systems are verified based on the requirements, including the real-time synchronization of "man, machine, material, and method" and the transient simulation in real time. The numerical indicators of the integrated system are verified, including the model completeness and synchronization timeliness.

1. Introduction

Intelligent manufacturing is a key initiative in today's major industrial countries to help companies gain competitive advantages. Regardless of the efforts of China's "Made in China 2025" initiative, the "Industrial Internet" in the United States, and Germany's "Industry 4.0" to promote the interaction and fusion between the virtual and the physical world, Cyber-physical systems (CPS) remain the core of intelligent manufacturing [1]. At present, the new generation of information technology, such as the Internet of Things (IoT), cloud computing, big data, and artificial intelligence, is infiltrating the traditional manufacturing industry with unprecedented depth, breadth, and granularity [2]. Therefore, the fusion of virtual and real world is an inevitable trend in the convergence of information technology and the manufacturing industry [3].

Digital Twin (DT) technology is a critical enabling technology of CPS, which provides a novel, feasible, and clear implementation path for the realization of virtual-real fusion by using digital technology [4]. Grieves introduced the concept of "virtual digital representation equivalent to physical products" in a product lifecycle management course in 2003 [5]. Later, this concept was named DT in 2011 [6], and it has garnered significant attention from researchers in various fields [7]. Applying DT to realize smart production and a smart shop floor is a major focus in manufacturing. In addition to a physical shop floor in the real world, people expect to have a "twin" shop floor in the virtual world, i.e., a shop-floor digital twin (SDT) or shop-floor digital twin model, because the SDT can be designed to monitor current conditions in real-time, track information from the past when needed, and assist in operational decision-making for the future [8,9].

Because the cost of trial-and-error can be greatly reduced by making full use of virtual space, constructing a cyber-physical production system (CPPS) is a reliable means of improving production efficiency, optimizing resource allocation, and reducing production costs. The SDT is a subsystem of CPPS, and as such, determining the best way to construct

* Corresponding author.
E-mail address: zhuangdavid@bit.edu.cn (C. Zhuang).

https://doi.org/10.1016/j.jmsy.2021.05.004
Received 7 December 2020; Received in revised form 10 April 2021; Accepted 10 May 2021
Available online 21 May 2021

and apply the SDT has always been a focus of researchers. Many enterprises and researchers have carried out relevant theoretical research and implementation of application cases [10]. Many scholars have focused on the theory and framework of modeling and applying the SDT to achieve real-time monitoring, human-robot collaboration and interaction, and production planning [11]. Leading companies such as Siemens, GE, Microsoft and Dassault were committed to developing a powerful platform to help users construct 3D scenes or simulation environments [10–12]. A shop floor is a large complex system that involves many elements, including man, machine, materials, methods, and environment, and many processes such as the technical flow, business process, logistics, and control flow. However, how to construct and develop the SDT system in a hierarchical, structured and modularized manner from a system perspective is seldom discussed, which easily leads to problems such as low reuse rate of the system blocks, lack of scalability, and high upgradation and maintenance costs. Because of this, a systematic construction method for the SDT is urgently needed.

The remainder of this paper is organized as follows. Section 2 summarizes the state-of-the-art of DT applications on the shop floor. In Section 3, the implementation process of constructing an SDT based on the MBSE is illustrated. The problem domain model of the SDT is discussed in Section 4 based on the modeling language SysML, modeling method "MagicGrid," and "V model" of systems engineering. The solution domain model is presented in Section 5 for three subsystems: the visualization system, synchronization system, and simulation system. In Section 6, these subsystems are developed and integrated for a real NC machining shop floor in the implementation domain. The contributions of the paper and the author's conclusions are presented in Section 7.

2. Overview of DT applications on the shop floor

From the published literature found in the Web of Science, Engineering Village and CNKI databases using the search terms digital twin/digital twins and digital twin shop floor, combined with the DT solution proposed by enterprises such as Siemens, Beckoff, Microsoft, Dassault, and GE, it is found that many scholars and manufacturing enterprises have made great efforts to apply DT technology in the production stage to realize a smart shop floor by constructing a CPPS, thereby improving production efficiency and quality, optimizing resource allocation, shortening the production cycle, and reducing the production energy consumption [13]. For example, Tao introduced the concept of the digital twin shop floor, which aims to achieve the optimal production and control of shop-floor through two-way real-time mapping and real-time interaction between a physical shop floor and a virtual shop floor [14]. Sun et al. investigated a DT-driven assembly-commissioning approach for high precision products with multidisciplinary coupling, introduced the corresponding framework and realized the assembly prediction and assembly-commissioning process optimization [15]. Leng et al. applied DT to achieve the rapid reconfiguration of automated manufacturing systems, so as to improve system performance while minimizing the overhead of the reconfiguration process [16]. Liu et al. presented a systematic framework to provide guidelines for the rapid configuration and easy runtime of DT-based CPPS by integrating technologies including DT modeling, resource registration/binding, event-driven distributed cooperation mechanisms and web technology [17]. Söderberg et al. proposed a DT for real-time geometry assurance based on a case of a sheet metal assembly station, and discussed how the DT concept allows for moving from mass production to individualized production [18]. Kong et al. proposed a data construction method to provide efficient and stable data support for the application of digital twin on the shop floor [19]. Xia et al. used DT to train deep reinforcement learning agents, so as to automate smart manufacturing systems under facile optimization environments [20].

SDT is comprehensive and accurate digital mapping of the corresponding physical shop floor, the construction of which is the premise of realizing digital twin shop floor and CPPS [21]. On the theoretical level, Tao et al. discussed the construction of SDT from four dimensions: geometry, physics, behavior and rule [22]. On this basis, other scholars believed it is also necessary to construct SDT from the data dimension and logic dimension [23,24]. Modoni et al. [11] discussed the technological panorama to realize the fully-synchronized shop floor, in which the digital counterpart of the shop floor, i.e., SDT, was composed of three blocks: *Component Logical Schema*, *Component Instances* and the *Historical Persisted Data*. At the application level, most scholars currently construct the SDT from the dimensions of geometry and logic. The geometric modeling of SDT is mainly achieved through 3D modeling software for model making, texture mapping, and animation production. On this basis, the model library is created to realize the construction of a 3D scene in virtual space for the physical shop floor [25]. The logic modeling of SDT is achieved mainly through abstracting and describing the internal shop-floor operation logic to analyze, synthesize, and optimize its structure and operating status. As a production system, discrete manufacturing shop floor such as NC machining shop floor and assembly shop floor for complex products is a typical discrete event dynamic system (DEDS). Its modeling methods mainly include fractal and informal modeling. Formal modeling includes the queuing network method [26], maximum algebra method, disturbance analysis method, and Petri net [27], while informal modeling includes the activity cycle diagram, flow chart, and object-oriented method [28]. Petri net is not only expressed in strict mathematical form, but can also be described by intuitive graphics. Therefore, it is suitable to describe the sequence, concurrency, conflict, synchronization, and other relations of DEDS, and is widely used in production system modeling. For example, Zhang [29] presented a hierarchical timed-colored Petri net model in an IoT environment, which is mapped with collected real-time data and can accurately reflect the operating status of shop floor through the change of the coloring token.

Apart from the aforementioned approaches of constructing the SDT, various scholars and leading companies have investigated the construction approaches of DTs in other fields. Schleich et al. [30] presented a reference model for product DT based on skin model shapes, which has the properties of scalability, interoperability, expansibility, and fidelity. Boschert et al. [31] emphasized that DT is the next wave of simulation and the benefits of DT would include the possibility of reuse in later lifecycle stages and the potential for using the complete set of product lifecycle information. Cheng et al. [32] introduced a DT-enhanced Industrial Internet (DT-II) reference framework that was illustrated from three levels: product lifecycle, intra-enterprise and inter-enterprise. Additionally, Siemens [33] applied three types of DTs – product DT, production DT and performance DT – throughout the product lifecycle based on the established MindSphere and Teamcenter platforms, aiming to simulate, predict, and optimize the product and production system before investing physical prototypes and assets. Beckoff [34] developed a TwinCAT Wind Framework software application to realize the condition monitoring, predictive maintenance and operation management of wind turbines. Microsoft [35] developed an IoT platform called Azure Digital Twins that enabled the design and construction of comprehensive digital models of entire environments, such as physics-based digital twins and infrastructure digital twins.

In conclusion, most of the academic research on the construction of SDT and DTs aimed to provide the guidelines of how to comprehensively describe a DT at the theoretical level. In the industry, to help users to construct SDT and DTs, leading companies such as Siemens and Microsoft provided platforms that consisted of general models, including geometry models of commonly used production elements, simulation models and logic models. However, a shop floor is typically a complex system that involves many elements, many processes and interrelated coupling among various components. The composition of elements, the simulation model, logic model and business process model on the different shop floor is diverse. Therefore, an important focus is how to realize the rapid system development of an SDT in a hierarchical, structured and modularized manner for different application scenes.

This article aims to provide an MBSE-based modular development solution for constructing an SDT that has its own customized requirements. Not only can the errors be avoided in the top-level system design stage through clear requirements description and intuitive design model diagrams, but a normalized and standardized model library can be formed. When creating an SDT model with similar functional modules, we can reuse the modules and realize the in-depth custom design according to the actual conditions of different shop floors to reduce software development time.

In addition, the SDT can be applied to assist decision-making for the operation optimization of the physical shop floor, and online prediction is one of the most important functions. Currently, there are two kinds of forecasting approach for shop-floor operating status. One is based on the established shop-floor operation logic model through offline simulation to analyze the impact of processing time, workpiece arrival, and other dynamic variables on the shop-floor performance indicators, such as completion time and delivery delay cost [36,37]. The other is to explore the evolution rules of the shop-floor performance through big data analysis. For example, for a multi-machine, multi-product, and order-oriented manufacturing shop floor, Ioannou et al. discussed dynamic prediction and updating methods for completion time, both of which can be incorporated into the ERP/MRP. One is based on the iterative decomposition algorithm, and the other is transforming it into an equivalent longest path problem. The effectiveness of the presented methods was verified through some examples [38]. Berling et al. used the random demand rate to solve the delay problem in the distributed system instead of random lead time demand. The numerical results showed that the presented methods were better than the existing methods, and the batch size and service level have great influence on the delay. On this basis, the authors were able to make predictions of the completion time in different supply chains [39]. In the current research on shop-floor operation prediction, the simulation method is mainly based on the manual initial system configuration and setting, and the simulation result is output according to the input. There is always a time difference between the simulation time and the actual system running time, thus the results obtained obviously lag behind. Prediction approaches driven by big data rely heavily on sufficient historical data. Nevertheless, obtaining a large quantity of historical data is not easy for many practical application scenarios. Moreover, a prediction approach driven by big data is an ideal steady-state prediction. In addition to steady-state prediction, it is necessary to achieve real-time transient prediction when the shop-floor operating status rapidly changes, which is also one of the core functions of SDT.

3. Implementation process of constructing SDT based on MBSE

A shop floor is a large complex system involving many elements, such as workers, machines, materials, methods, the environment, and many processes, such as logistics, control flow, technical flow, and business processes. To address the existing problems in the current research on the construction and application of SDT, from the system perspective, we propose an MBSE methodology-based construction method of SDT. Systems engineering (SE) [40] is an organizational management technology that includes text-based SE and model-based SE (MBSE). MBSE [41], which is an evolution of traditional text-based SE, descripts and expresses a system in a digital and graphical manner, rather than in text. MBSE is easy to understand and prevents making mistakes, which greatly improves efficiency. By introducing low costs, high efficiency and clear requirements into SE, MBSE frees people from doing manual labor so they can perform more creative work.

The most popular model in SE is the “V model” lifecycle [42], as shown in Fig. 1. The “V-model” summarizes the workflow of SE, which uses the top-down mode for forward design and bottom-up for reverse verification. The MBSE methodology has three core components: modeling language, modeling method, and modeling tools [43].

System Modeling Language (SysML) is adopted in this research, which is an object-oriented paradigm extended from the sub-set of UML2.0. It can visualize a system’s structure, behavior, requirements, and mathematical model, and can be applied for SE modeling of any scale containing complex elements such as hardware, personnel, and equipment [43]. The SysML grammar consists of three types of nine diagrams, as shown in Fig. 2. The Behavior diagram includes four subclasses: Activity Diagram, Sequence Diagram, State Machine Diagram, and Use Case Diagram. The Structure diagram includes: Block Definition Diagram, Internal Block Diagram, and Package Diagram. The Parameter Diagram is the subclass of the Internal Block Diagram, while the Requirements Diagram is a different type of diagram.

In terms of modeling method, we use MagicGrid [44], a new-generation modeling method of MBSE. It is independent of the modeling tool and fully compatible with SysML. The MagicGrid approach defines the implementation process of constructing a system model as three domains, which are: problem domain, solution domain,

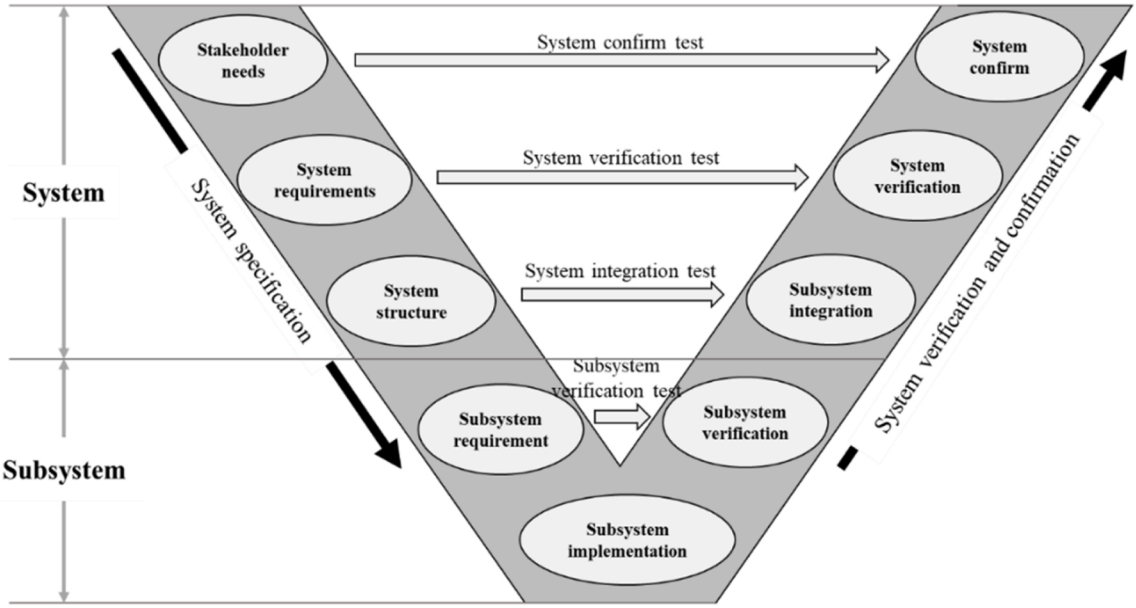

Fig. 1. “V model” lifecycle of systems engineering.

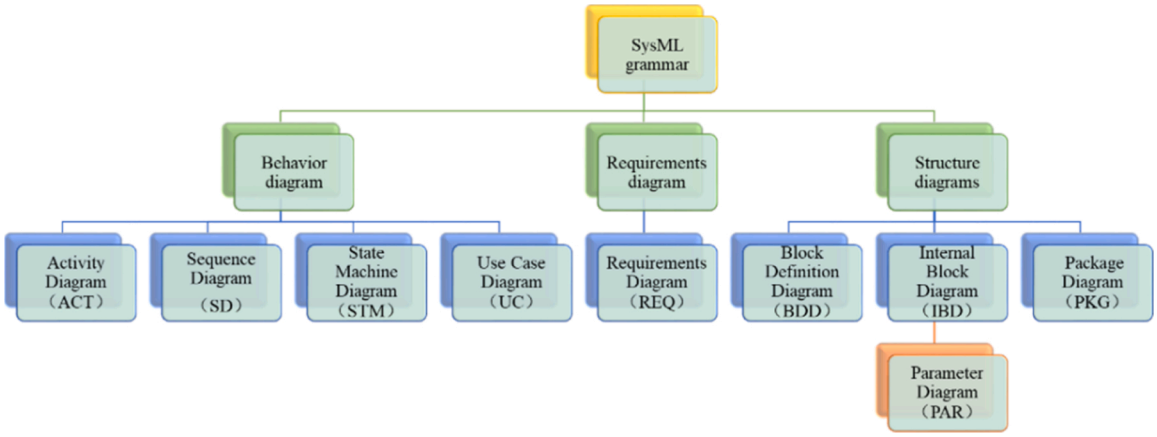

Fig. 2. SysML grammar.

and implementation domain. The problem domain is the clarification process of the stakeholder needs; the solution domain is the definition process of the system architecture; and the implementation domain is the detailed realization process of the branch disciplines. The problem domain is composed of two parts: black box and white box. This approach means that the system is first viewed from the black box perspective, and then the decomposed from the white box perspective to make the System of Interest (SoI) clear and accurately define the problem domain. The solution domain is composed of many internal rows, which means the system's solution architecture can have several levels. The implementation domain is the detailed design and development of the branch disciplines. Each domain consists of four modeling dimensions that should be considered and implemented: requirements, behavior, structure, and parameters.

The modeling tools of MBSE are often related to the modeling method, such as the aptly named software Camero Systems Modeler. It was developed to match the MagicGrid by No Magic Corporation.

Based on the above analysis, the implementation process of constructing an SDT based on the MBSE is established, as shown in Fig. 3.

The process is as follows. First, establish the problem domain model of SDT. In the black box stage, define the system requirements of SDT, analyze the external environment of the SDT, build use cases, and determine the parameters of measurement of effectiveness (MoEs). In the white box stage, complete the functional analysis of the SDT, sort out the interactions among the three subsystems of the SDT, including the visualization system, synchronization system, and simulation system, and then refine the MoEs parameters. Second, the solution domain model is constructed for the three subsystems, which is the main body of SDT. For each subsystem, based on system requirements, several iterations are carried out for the three dimensions of structure, behavior, and parameters at the subsystem level and component level. Finally, in the implementation domain, the SDT is developed and implemented for a real shop floor to verify the proposed MBSE method. For the visualization system, the scheme of combining commercial software is given. For the synchronization system, the detailed technical scheme covering four aspects of "man, machine, material, and method" and the realization scheme of data transmission is included. We propose a continuous transient simulation based on real-time data for the simulation system

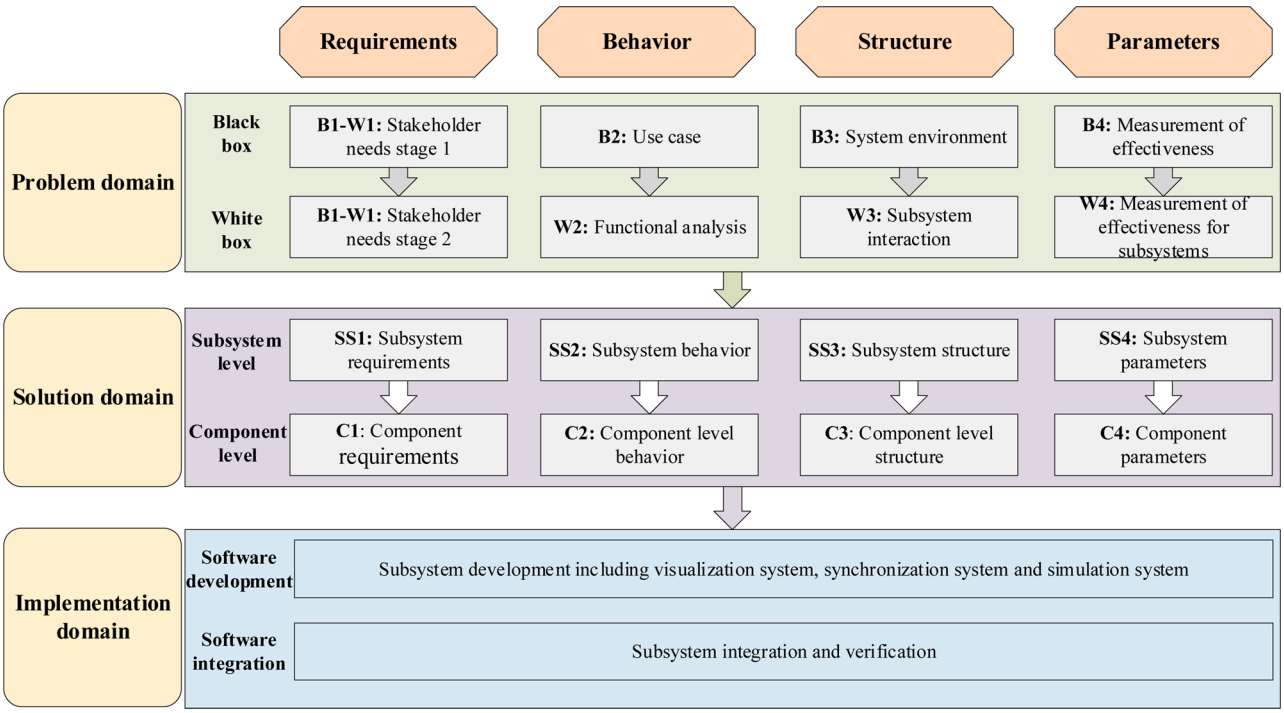

Fig. 3. Implementation process of constructing an SDT based on the MBSE.

based on event-driven DEDS technology. In the system integration and verification phase, several interfaces are displayed to verify the functions of the SDT based on the requirements. The numerical indicators of the SDT are verified, including the model completeness and synchronization timeliness.

4. Problem domain

The purpose of the problem domain is to analyze the stakeholder needs of the SDT and to preliminarily optimize the model, thereby identifying the problems that need to be solved by the SDT.

4.1. Black box process

4.1.1. B1-W1: stakeholder needs stage 1 of the SDT

The B1-W1 cells represent the collection of stakeholder needs of the SDT. Stage 1 of B1-W1 is performed in the black box stage. We use the Requirement Diagram of SysML to model stakeholder needs. The stakeholder is the shop-floor manager. The code, content, and relationship of the stakeholder needs in SDT are shown in Fig. 4.

4.1.2. B3: system environment of SDT

The B3 cell represents the modeling of the black box for the structural characteristics of SoI, i.e., identifying the external environment of the SDT in the CPPS environment. In the phase of confirming the system environment, the shop-floor manager and physical shop floor are introduced as the external environment of SDT. The external environment is modeled using the Block Definition Diagram of SysML, as shown in Fig. 5.

4.1.3. B2: use case of SDT

The B2 cell represents the modeling of black box for the behavioral characteristics of SoI. We use the Use Case Diagram of SysML to model the system use case. As shown in Fig. 6, the shop-floor manager wants to see the synchronization of the virtual and physical space from the SDT and obtain simulation data to make decisions. Thus, two basic cases arise, namely, *See Synchronous Scene* and *Get Simulation Data*. For the *See Synchronization Scene*, the shop-floor manager first wants to see the panorama of the physical shop floor in the virtual space, and then the synchronization of the SDT and physical shop floor. Therefore, the use case of *See Synchronous Scene* has two connotative use cases, including *See the Whole Scene* and *See the Synchronous Action*.

The use case scenario is the description and refinement of the use case. Taking the *See the Synchronous Action* as an example, the use case scenario is modeled using the Activity Diagram of SysML, as shown in Fig. 7.

Shop-floor data collected in the physical shop floor by using data acquisition equipment are sent to *Transit Database*. The next data acquisition and transmission takes place after the time of *Physical Shop floor: Data Acquisition Rate*. After the data package is sent to *Transit Database* from the physical shop floor through the object flow, the *Transit Database* immediately stores the data package. Afterwards, *Update Scenarios Action* reads the data package from the *Transit Database* to complete the update of the scene. If no special conditions exist, after the

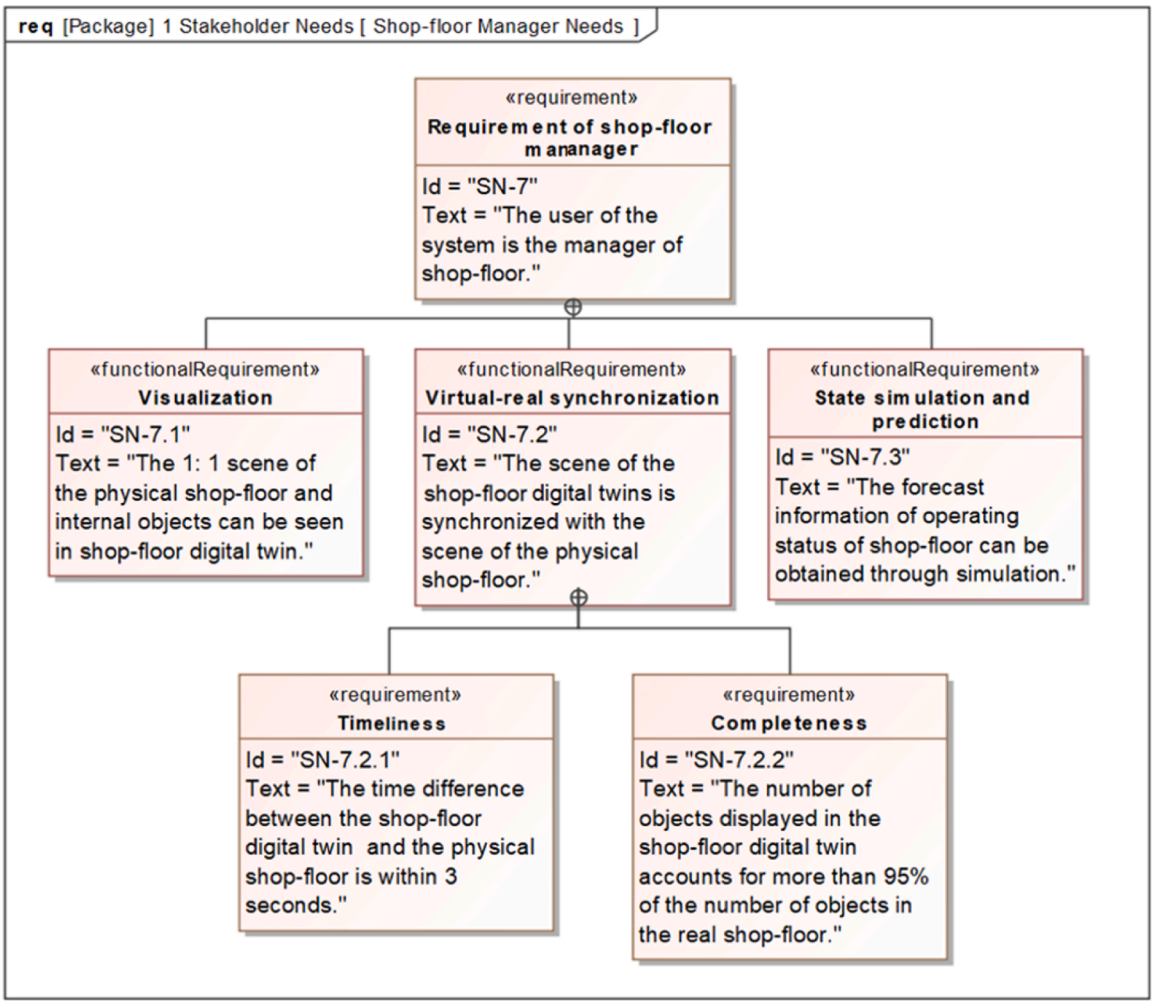

Fig. 4. B1-W1: Requirement Diagram of stakeholder needs.

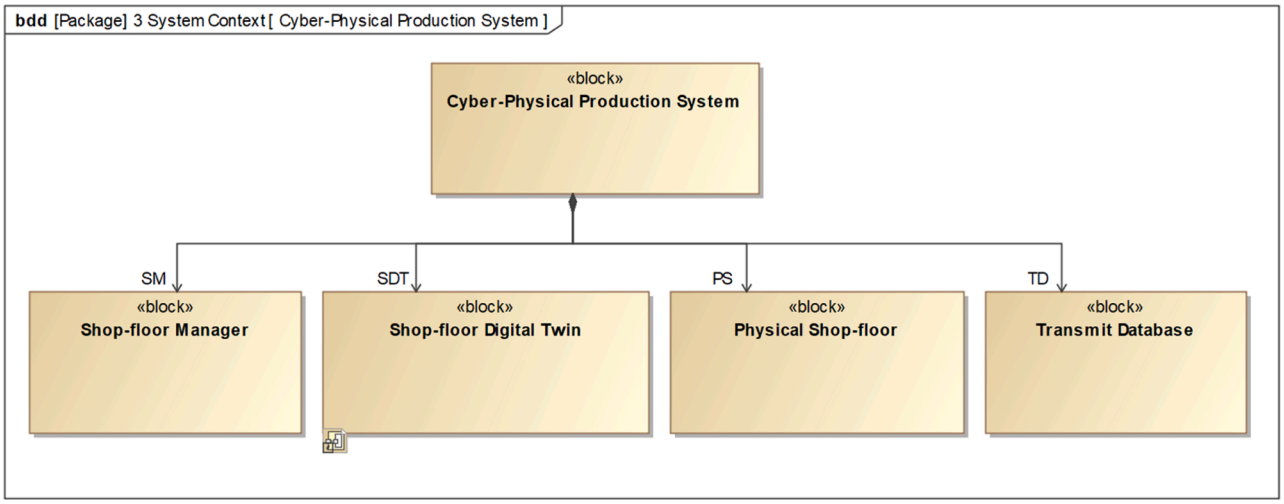

Fig. 5. Block Definition Diagram of system environment of SDT.

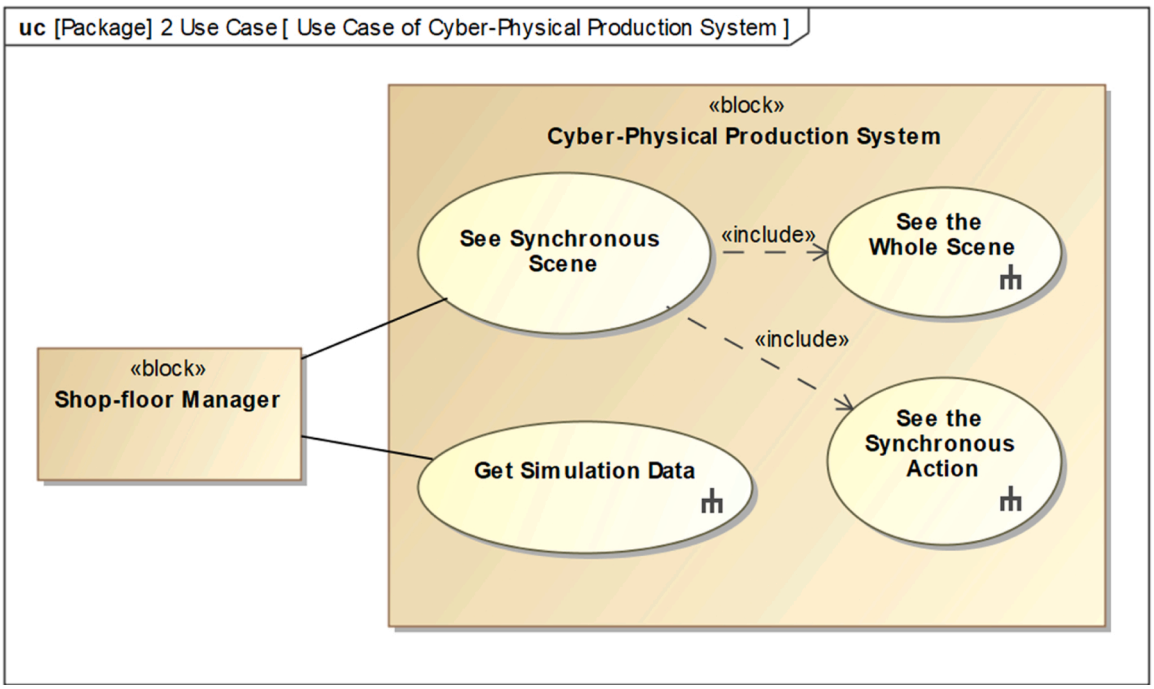

Fig. 6. Use Case Diagram of use case of SDT.

time expires for *Shop-floor Digital Twin: Data Reading Rate*, the next data reading, parsing, assignment, and updating of the scene actions will be performed. Otherwise, when the *Close* condition is true, the use case of the behavior synchronization ends.

4.1.4. B4: MoEs of the SDT

The B4 cell represents the refinement of the non-functional stakeholder needs by MoEs models the parameter characteristics of SoI numerically. The MoEs is a traditional terminology widely used in SE to describe the system's ability to perform tasks in a specific environment. In the problem domain model, MoEs can act as high-level key performance indicators and can be automatically checked in the solution domain model. We use Block Definition Diagram to model MoEs, as shown in Fig. 8. Non-functional stakeholder needs are characterized as system MoEs and expressed as value attributes of *MoEs Holder* block. By inheriting *MoEs Holder*, the SDT obtains two value attributes, including *Complete Rate* and *Time Difference*.

4.1.5. Organization mode of black box process model

We use the Package Diagram to express the organizational mode of the model elements. The Package Diagram of the black box process model is shown in Fig. 9. The model is organized according to the four modeling dimensions defined by the MagicGrid approach. *SDT_Black Box* is the top package of the black box process, with four secondary packages. Of these, the "*1 Stakeholder Needs*" package is to store stakeholder needs. The "*2 Use Case*" package stores use cases and actions in the Activity Diagram, as well as the type of action, i.e., activities. The "*3 System Context*" package stores the blocks involved in the system environment. MoEs are stored in the "*4 Measurement of Effectiveness*" package.

4.2. White box process

4.2.1. W2: function analysis of the SDT

The W2 cell represents the breakdown of top-level functions

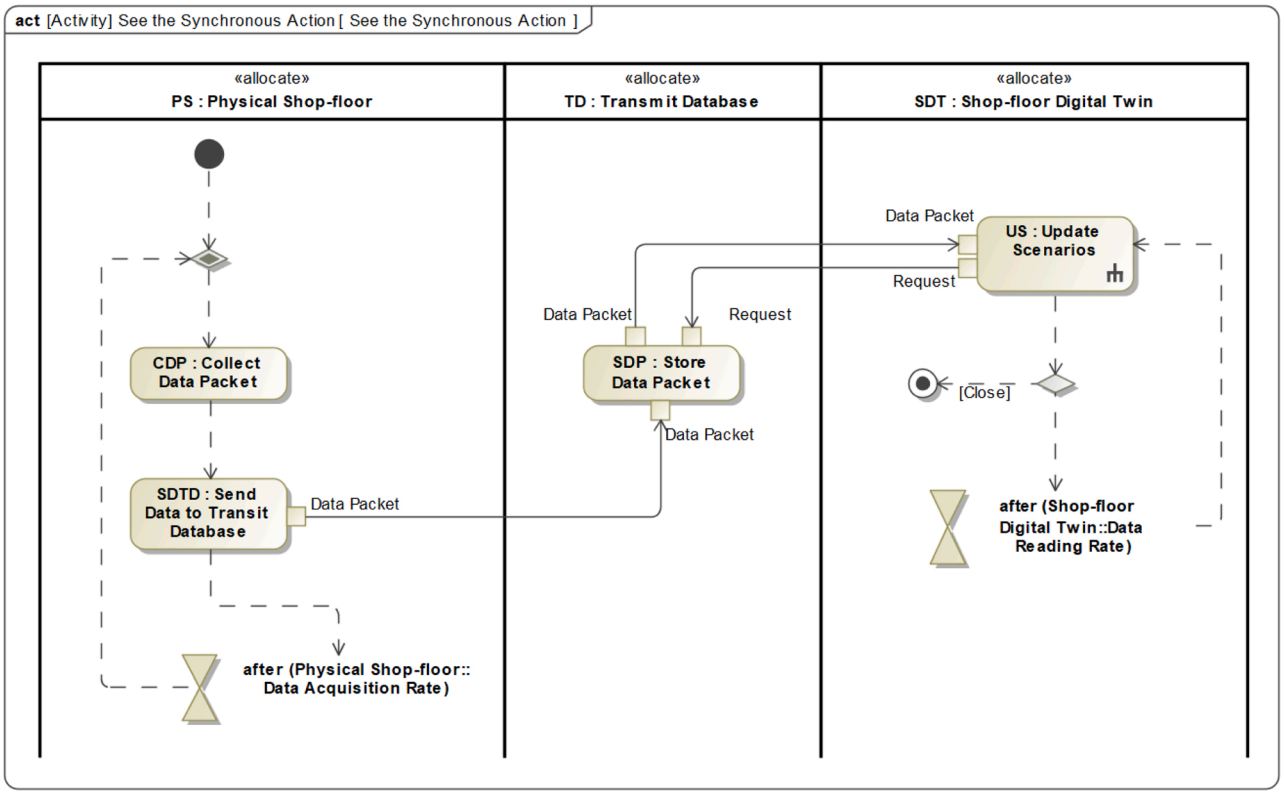

Fig. 7. Activity Diagram of the use case of *See the Synchronous Action*.

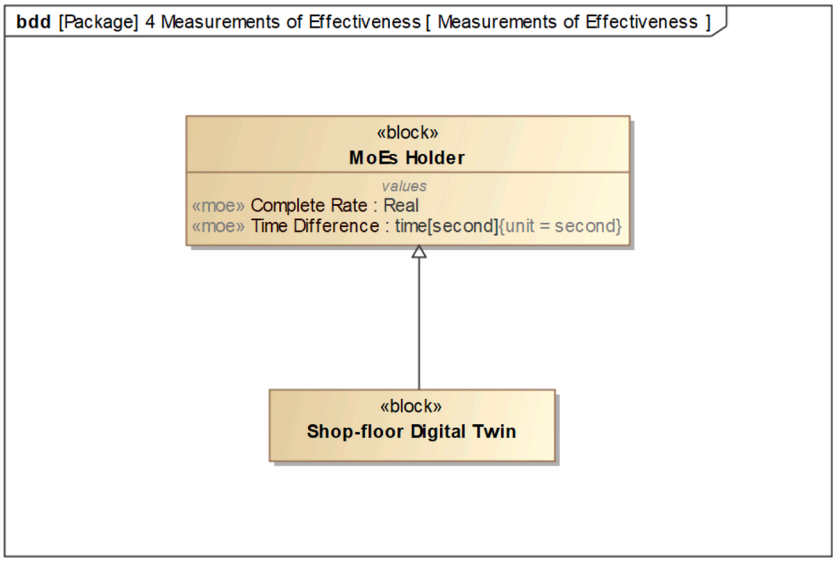

Fig. 8. Block Definition Diagram of MoEs of the SDT.

identified at the black box stage and specifies more detailed system behavior. As shown in Fig. 10, the Activity Diagram is used to continuously refine the behavioral characteristics of the SDT, including use cases and activities.

Fig. 11 is an Activity Diagram for analyzing the functions of the *Update Scenarios* in Fig. 7. If the Send Request Action "*Request*" sends a signal instance of type Request, the flow ends. If the Response Receiving Action "*Response*" receives a signal instance of type Response, the activity continues. The Response signal is *decoded* and then the *Test Effectiveness* is performed. If the data are valid, *Analysis Data*, *Assign Value*, and then *Update Position and Posture* are performed. If the data are invalid, the activity ends.

4.2.2. *W3: subsystem interactions*

The W3 cell represents the identification of the subsystems based on the functional analysis and the interactions between the subsystems. In phase W3, SoI transfers from the SDT to its subsystem, and the inputs and outputs of the subsystem are identified. (Since there are multiple SoIs, the same model structure is established for each SoI to make them easy to read.)

Fig. 12 shows the Internal Block Diagram of the SDT. It has three subsystems: visualization system, synchronization system, and

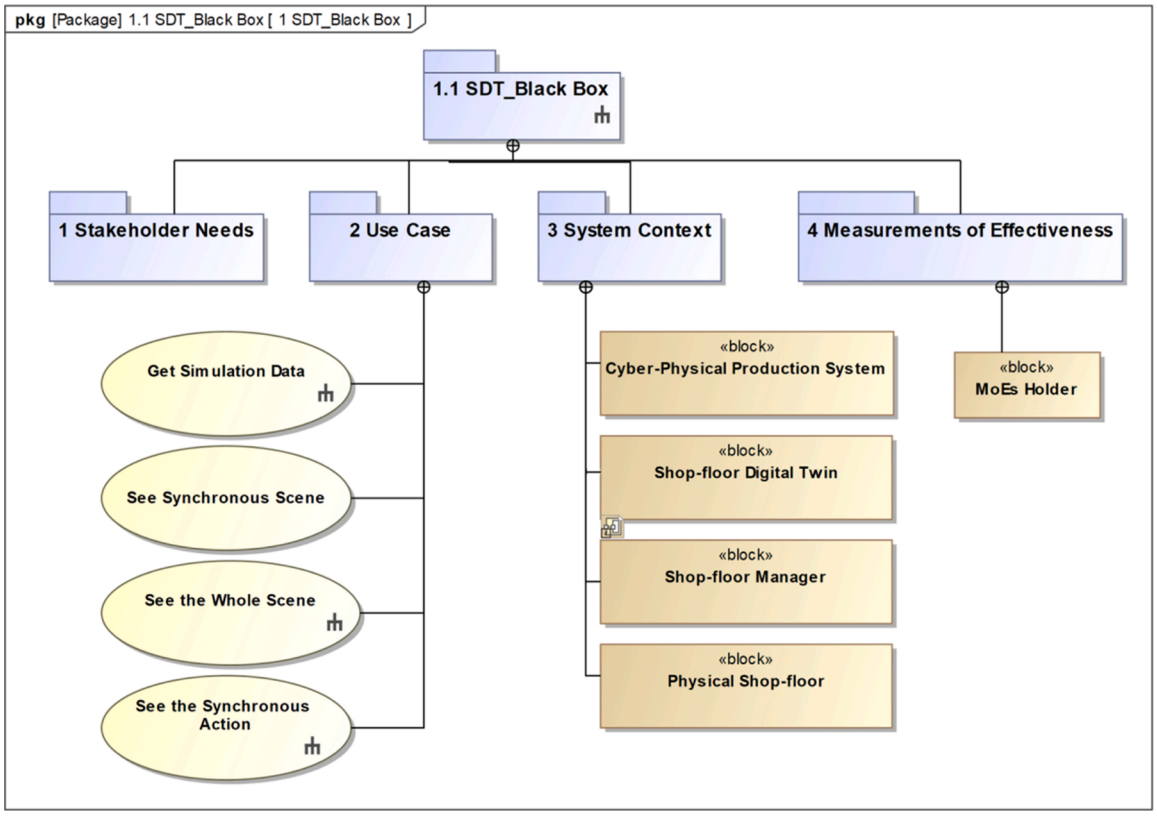

Fig. 9. Package Diagram of black box process model.

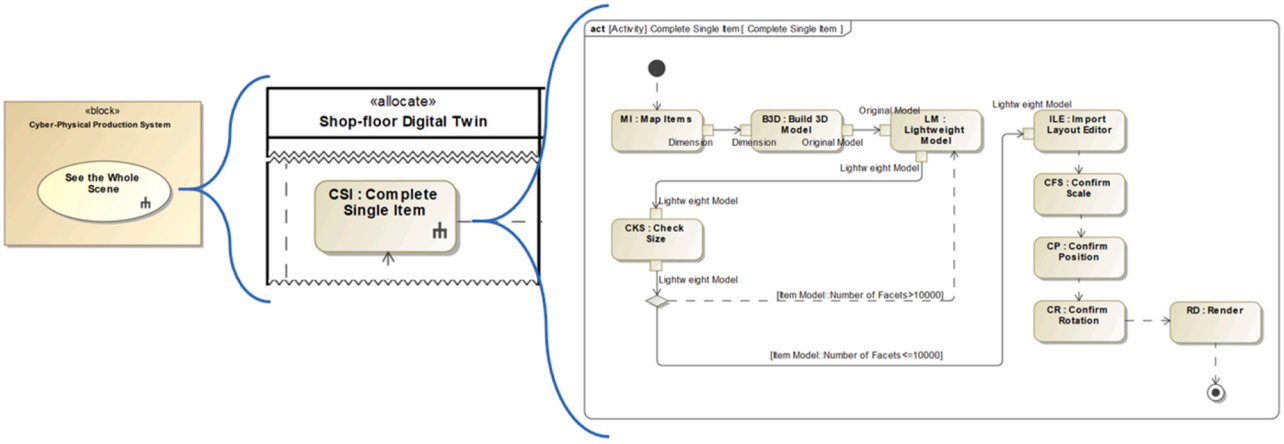

Fig. 10. Continuous refinement of behavior characteristics of SDT.

simulation system. Interaction between the subsystems relies on connectors and ports. The input ports of the SDT are *Dimensional Data Port* and *Remote Data Transfer Port*, while the output ports are *Synchronized Scene Port* and *Operational Data Port*. The interaction between subsystems is shown in Fig. 12. The visualization system is modeled with dimensional data, and the synchronization system and simulation system use remote data to synchronize and simulate. The visualization system is also responsible for the output and display of synchronized scenes and simulation data.

After the Internal Block Diagram is created, the relevant blocks are declared using the Block Definition Diagram, as shown in Fig. 13. The subsystems of SDT and their internal interfaces are declared, and the flow properties that allow the flow in the interface block are supplemented.

4.2.3. W4: MoEs of the subsystems of the SDT

Similar to section 4.1.4, the MoEs are specified for each subsystem. Constraint blocks are built using Block Definition Diagram to refine MoEs. As shown in Fig. 14, the *Cyber-Physical Production System* has three constraining blocks: (identified with the keyword < constraint >>), including *Total Time*, *Complete Rate*, and *Time Effectiveness*. The formulation of calculating the total time is built in the *Total Time* constraint block, which can be used to measure the time effectiveness requirements of stakeholder needs with *Time Effectiveness*. The *Complete Rate* constraint block establishes a computational formulation for the rate of model integrity, which can be used to measure the completeness requirements of stakeholder needs. The MoEs of subsystems refine the MoEs of large systems.

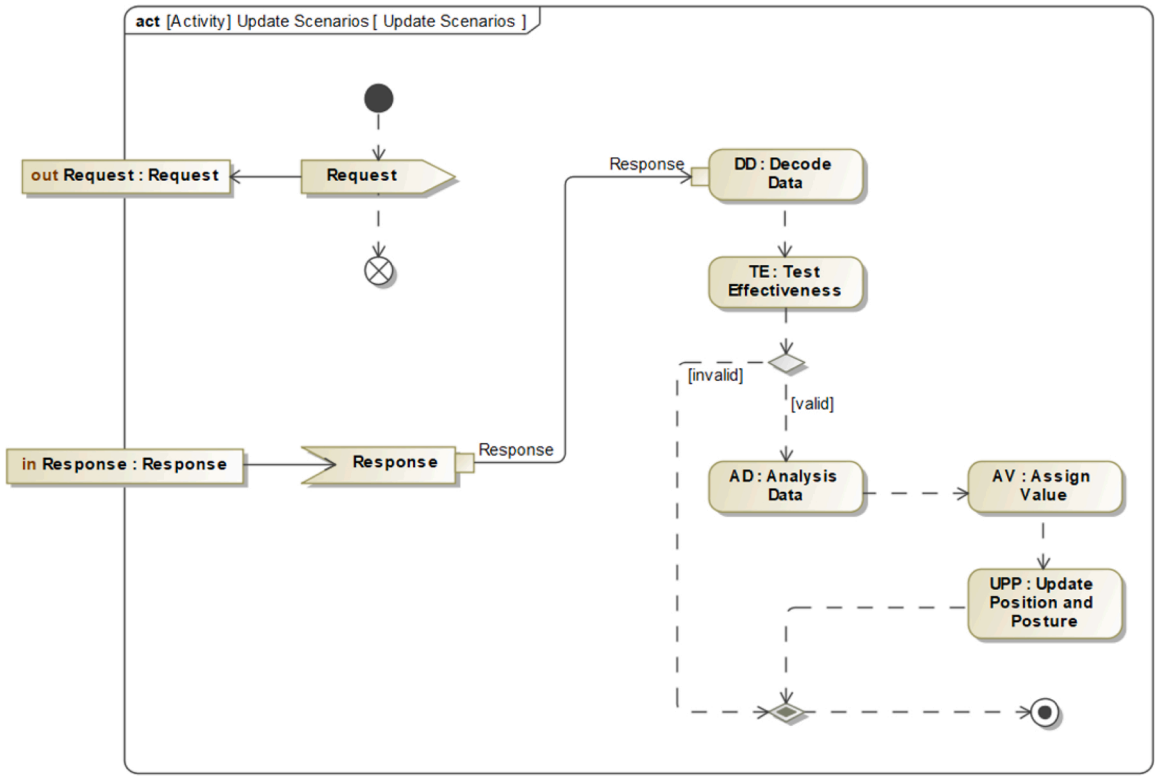

Fig. 11. Activity Diagram for the function analysis of *Update Scenarios*.

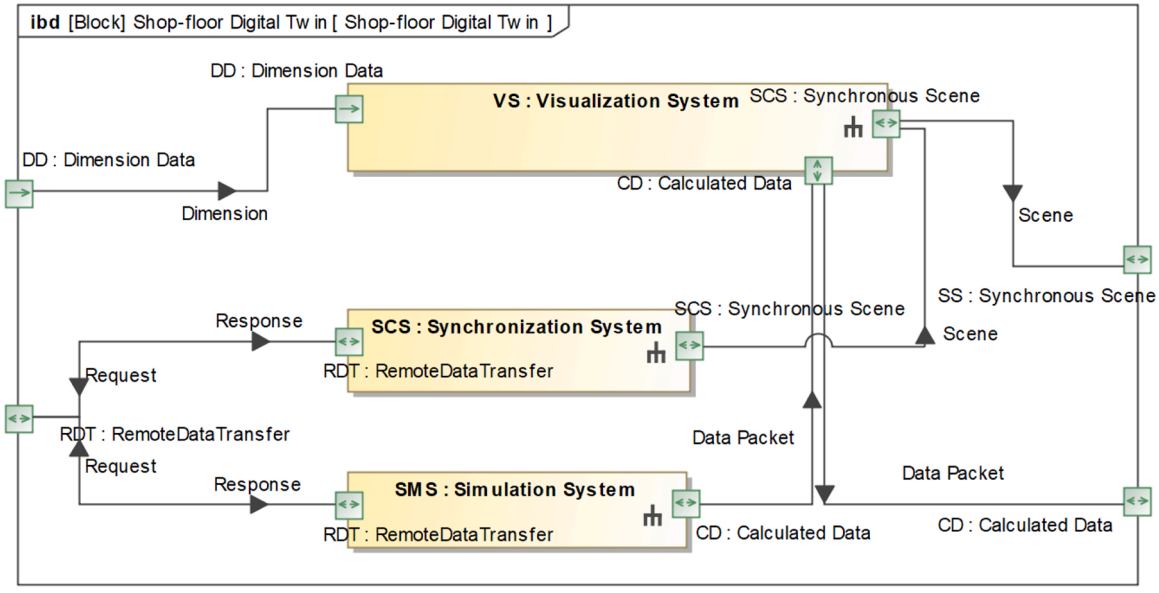

Fig. 12. Internal Block Diagram of SDT.

4.2.4. *B1-W1: stakeholder needs stage 2 of SDT*

In order to quickly locate the affected model elements when requirements change, each requirement should be refined by one or more elements. Therefore, Refine Matrix is applied to establish a retrospective relationship between model elements and stakeholder needs. As shown in Fig. 15, the solid line arrow (↙) in the diagram represents that the model element in the matrix column (such as activity, block, constraint block, and value attribute) is an explicit refinement of the requirement element in the matrix row, which is identified by the keyword “Refine”. Dotted arrows indicate that elements in the matrix column implicitly refine the requirements in the matrix row.

5. Solution domain

The visualization system, synchronization system, and simulation system are designed in the solution domain. This section uses a synchronous system as an example to show the detailed modeling process, which expresses the input and output of the synchronization system and its interactive interface.

5.1. *SS1: phase 1 of the synchronization system requirements*

The SS1 cell represents the requirements of a subsystem, which is

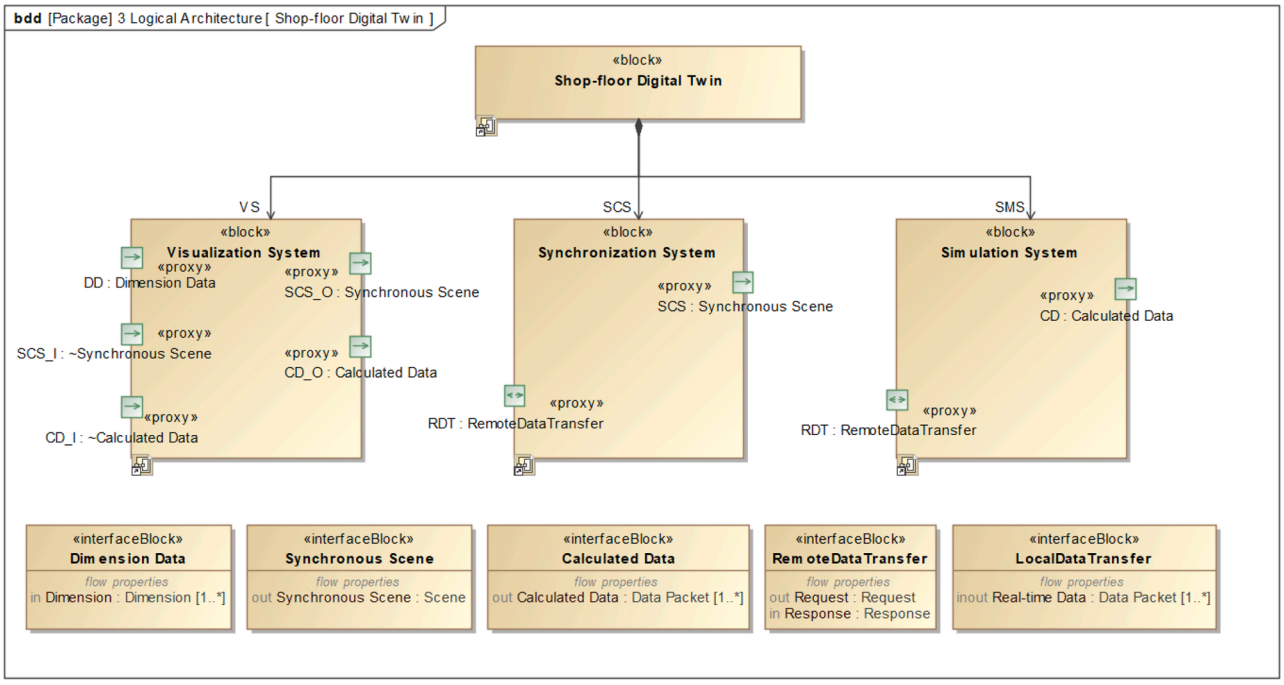

Fig. 13. Block Definition Diagram of SDT.

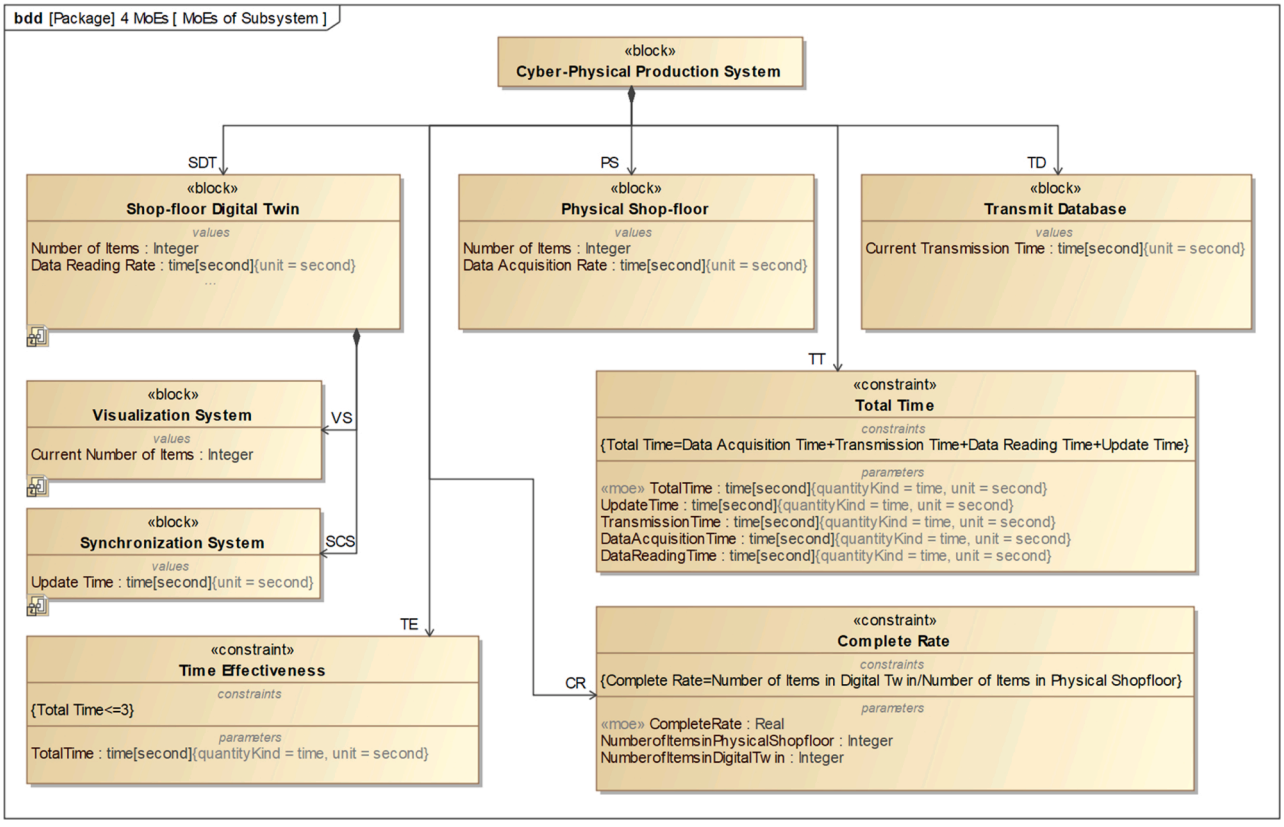

Fig. 14. MoEs of the subsystem of the SDT using Block Definition Diagram.

composed of two phases. In Phase 1, the subsystem requirements are decomposed based on the function requirements "virtual-real synchronization" of the stakeholder needs. As shown in Fig. 16, the synchronization system has a parent requirement named *Synchronization* and contains five sub-requirements, namely *Auto-Sync*, *Human Sync*, *Machine Sync*, *Material Sync*, and *Information Sync*. All of them are functional requirements. We establish a retrospective relationship between the requirements of the synchronization system and the model elements of the problem domain, as shown in Fig. 17.

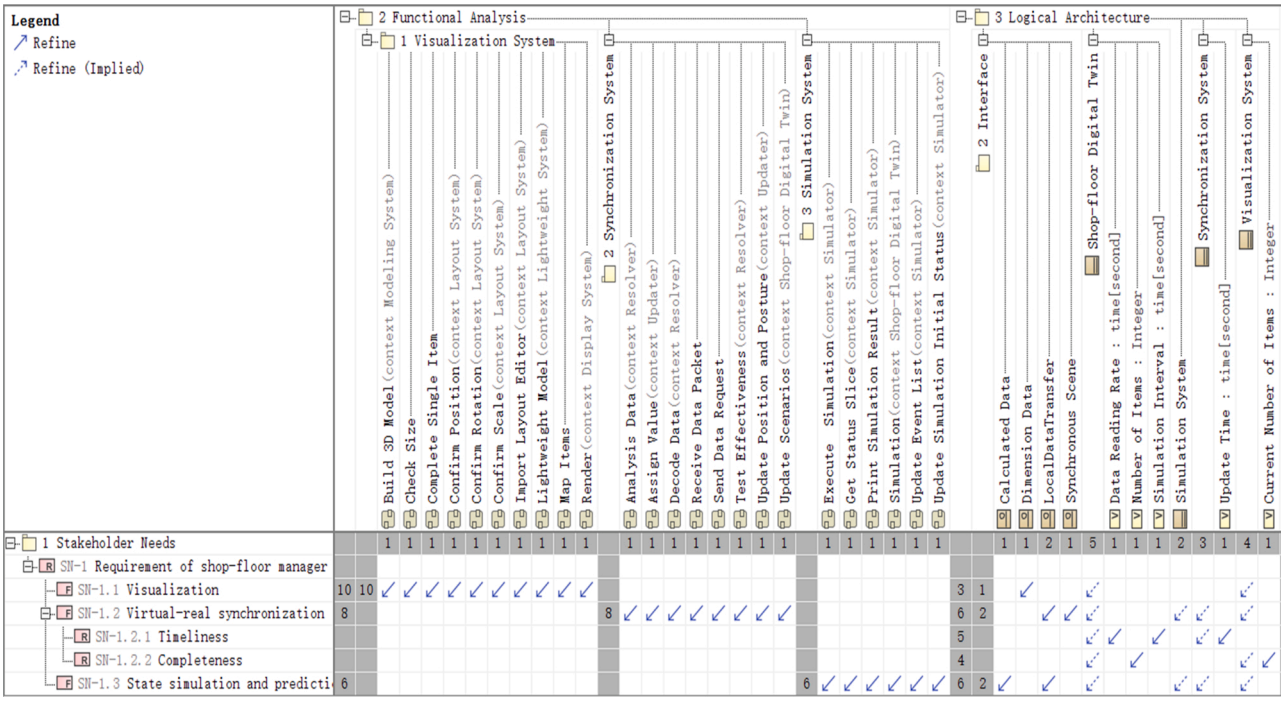

Fig. 15. Traceable refined matrix of model elements to stakeholder needs.

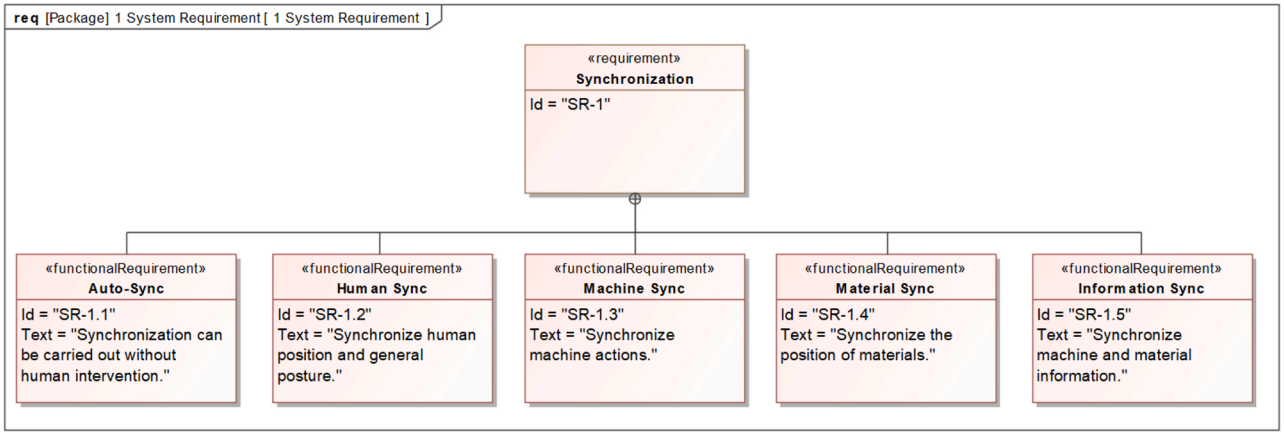

Fig. 16. Requirements of the synchronization system.

5.2. Synchronization system structure

5.2.1. SS3: subsystem level structure of the synchronization system

The SS3 cell represents the structure of the subsystem. We use the Internal Block Diagram of SysML to describe the internal structure of the synchronization system, as shown in Fig. 18. The synchronization system consists of three subsystems: *Reader*, *Resolver*, and *Updater*. *Reader* communicates with external systems (i.e., the data servers) via the RDT ports, including sending request signals (*Request*) and receiving response signals (*Response*), and outputs the response information to *Resolver* via the LDT ports. *Resolver* parses the data and outputs *Serialized Data* to *Updater* via the LDT port. After *Updater* brushes *Serialized Data* into the *Scene*, it outputs *Synchronous Scene* to the outside world (i.e., the layout system of the visualization system) via the SCS port.

To improve the design of the subsystem structure, the synchronization system and its three subsystems are further refined, as shown in Fig. 19. There are three subclasses of synchronization systems: *Human Sync System*, *Machine Sync System*, and *Material Sync System*. *Reader* has two subclasses, including *Database Reader* and *HTTP Reader*. *Resolver* has two subclasses, including *Database Resolver* and *HTTP Resolver*. There are four subclasses of *Updater*: *Material Updater*, *Machine Updater*, *Human Updater*, and *Information Updater*. Since both the equipment and material have more detailed information to synchronize, the *Machine Sync System* and the *Material Sync System* each have a component attribute called IU.

To define the relationship in the structure, the port types and flow objects between the structures are refined. As shown in Fig. 20, on the left side of the diagram, the remote data transmission interfaces consist of two categories: the *Database Transfer* interface and *HTTP Transfer* interface. The right side of the diagram shows that both *Response Json* and *Stream* belong to *Response*; both *Query SQL* and *Request Json* belong to *Request*; *Array* belongs to *Serialized Data*. *Response*, *Request*, and *Serialized Data* all belong to *Data Packet*.

5.2.2. C3: component level structure of synchronous system

The synchronization system is realized by three subclasses: *Human Sync System*, *Machine Sync System*, and *Material Sync System*. This section takes *Human Sync System* as an example to explain the design process of component level (C level) structure. As shown in Fig. 21, *Human Sync*

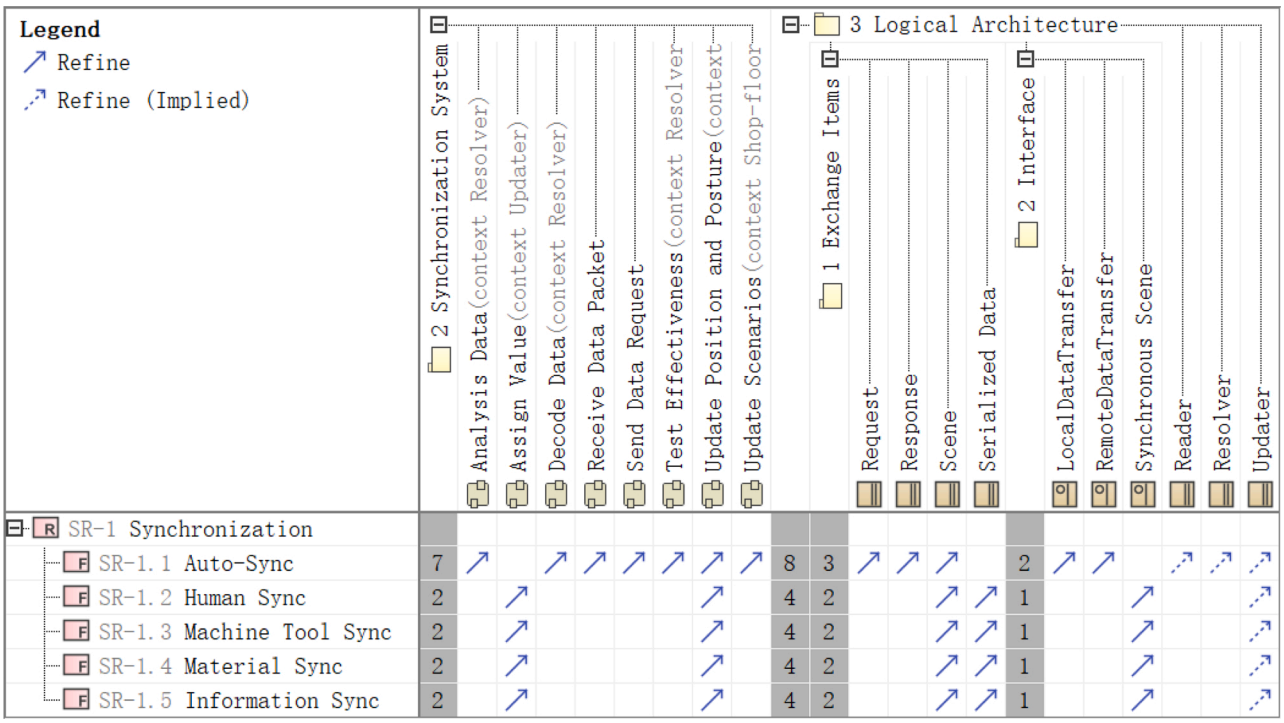

Fig. 17. Traceable refined matrix of the synchronization system requirements to the problem domain model elements.

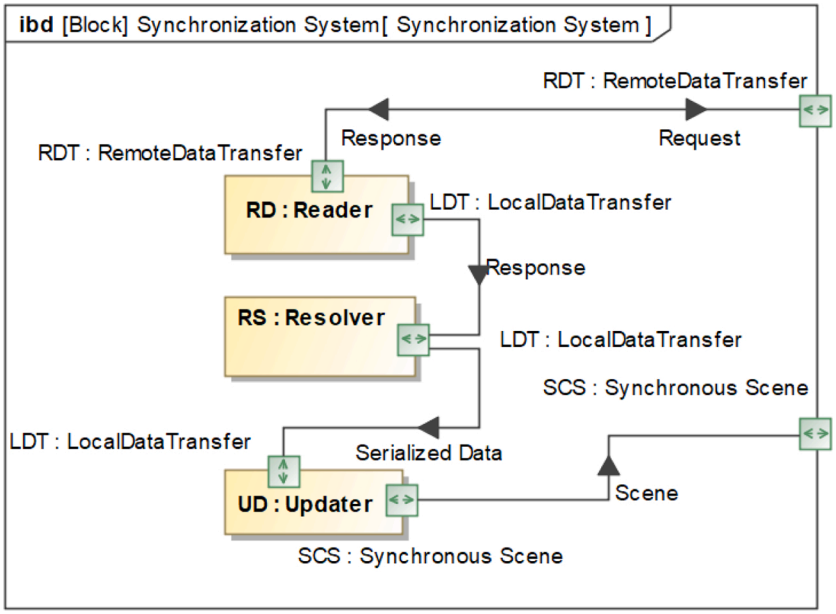

Fig. 18. Internal Block Diagram of the synchronization system.

System inherits the component attributes of the synchronization system. Human synchronization depends on the location data collected from the shop-floor IoT. The SDT is connected to the shop-floor IoT through the database, so *Reader* is specialized as a *Database Reader*, *Resolver* is specialized as *Database Resolver*, and *Updater* is specialized as *Human Updater*. The *Database Reader* transmits information to the external system (i.e., database server) through "DT" port, including sending the SQL query and receiving the Stream response. The type of "DT" port is *Database Transfer*. Then, *Database Reader* transmits the flow information to *Database Resolver* through the LDT port, the type of which is *Local Data Transfer*. *Database Resolver* converts the flow of information into an array and sends the array to the Human Updater through the LDT port. *Human Updater* outputs synchronized pictures to the outside of the system (i.e., the layout of the visualization system) via the SCS port. Most ports in the figure inherit from the parent class, which is identified by the symbol ˆ before the port name.

To model the behavioral characteristics of components in the synchronous system, it is necessary to declare the structures and attributes involved in component interactions. As shown in Fig. 22, *Shop-floor Object* has three value attributes: position, direction, and name. The type of location and direction are 3D coordinate value types (identified by keyword<< value Type>>) and the type of name is a string. Additionally, *Shop-floor Object* has an operation called *Update*, the parameter of which is *Data Body* of type *Array*. The purpose of this operation is to

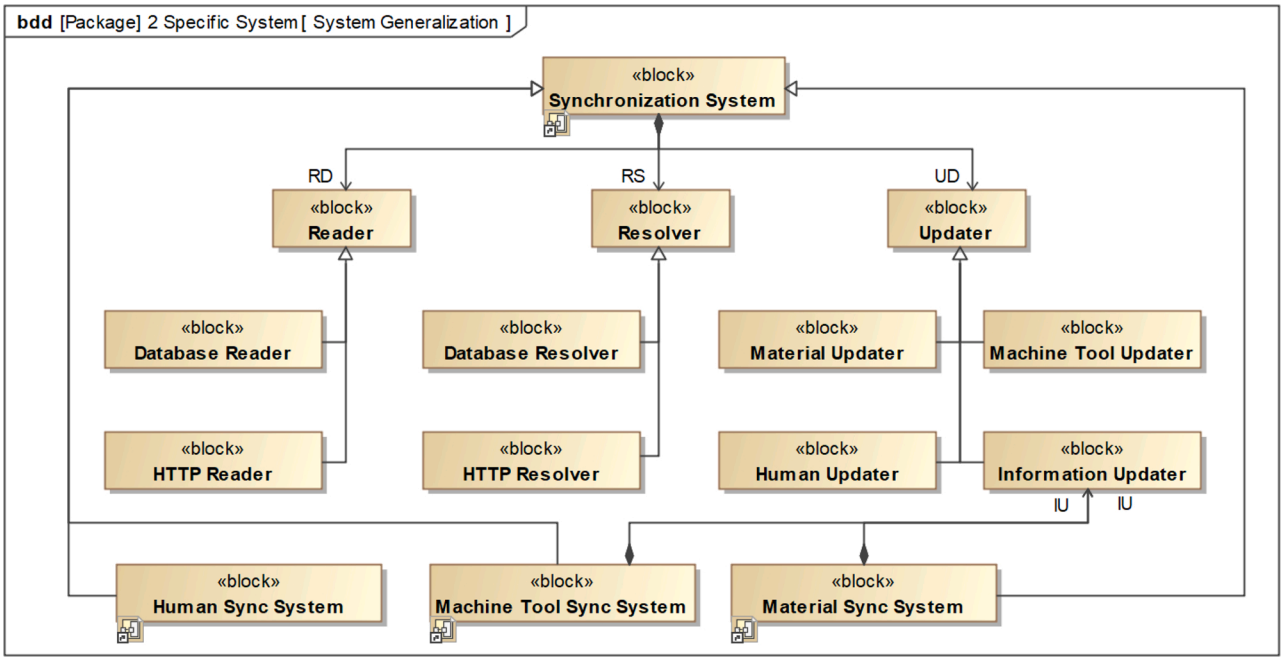

Fig. 19. Structure refinement of the synchronization system.

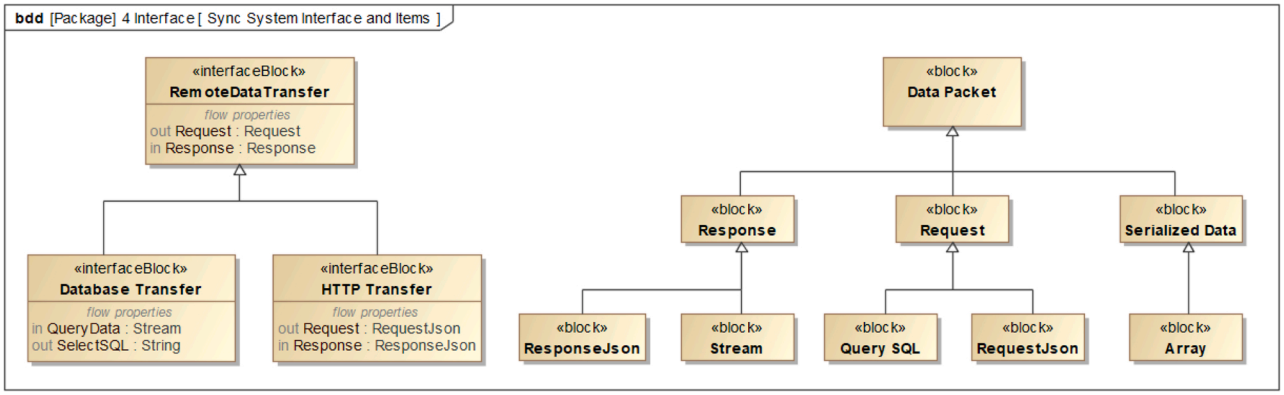

Fig. 20. Refinement for the port types and flow objects of the synchronization system.

extract the position, rotation, and name information from the external input data body and update them to its own value attributes. *Human*, *Information Board*, *Material*, and *Machine Tool* are inherited from *Shop-floor Object*. In addition to inheriting the value attributes and operations, the *Human* block has a state machine called *Human State* as a component attribute. The *Information Board* block has an information component attribute of type *Text*, and the multiplicity of the component attribute is greater than or equal to 1, which is identified by the symbol "[1..*]". The *Material* block has an ID value attribute of type *String*. *Machine Tool* has a component attribute *Part* and a value attribute ID, in which the type of *Part* is *Shop-floor Object* and the multiplicity is [1..*], and the type of ID is *String*. Moreover, both *Material* and *Machine* use *Information Board* as part of their attributes.

5.3. Synchronization system behavior

5.3.1. SS2: subsystem-level behavior of the synchronization system

We use the Activity Diagram to assign behavior characteristics to a structure, as shown in Fig. 23.

Declare the block involved and the behavior attributes it owns, i.e., operation, which is identified by the keyword << operations >>. Use a Block Definition Diagram to declare the operation of the block. As shown in Fig. 24, both *Database* Server and *HTTP* Server inherit the *Server* block, thus inheriting the *Execute Return* operation of the Server. *Database Server* rewrites this operation as Execute SQL and the *HTTP* Server rewrites it as *Execute Response*. Similar inheritance and rewrite relationships exist between the *Reader*, *Resolver* and *Updater* blocks and their subclasses. Overall, *Reader* has the operation of *Sending Data Requests* to the *Server* and sending data after receiving the return data package. *Resolver* has the operations of *Decoding data*, *Testing Effectiveness*, *Analysis Head*, and *Body*. *Updater* has *Search* for *All Shop-floor Objects* based on the data *Head* and *Assign Value* for attributes of objects using the data body. The parameters and their types for each operation are bracketed in the parameter list after the operation name. If the operation has a return value, the return value type is identified by a colon after the parameter list.

5.3.2. C2: component-level behavior of the synchronization system

Here, component-level behavioral modeling is discussed using the synchronization component as an example. A Sequence Diagram is a dynamic view of system behavior that describes the sequence of behaviors and events that occur over time. It is the precise description of behavior suitable for detailed system design. It can also output the object-oriented source code directly. Fig. 25 is the Sequence Diagram of

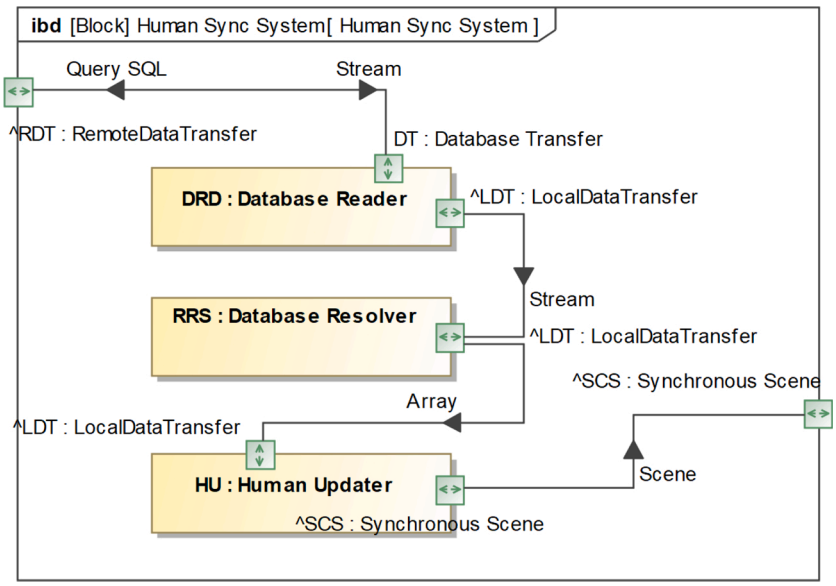

Fig. 21. Internal Block Diagram of Human Sync System.

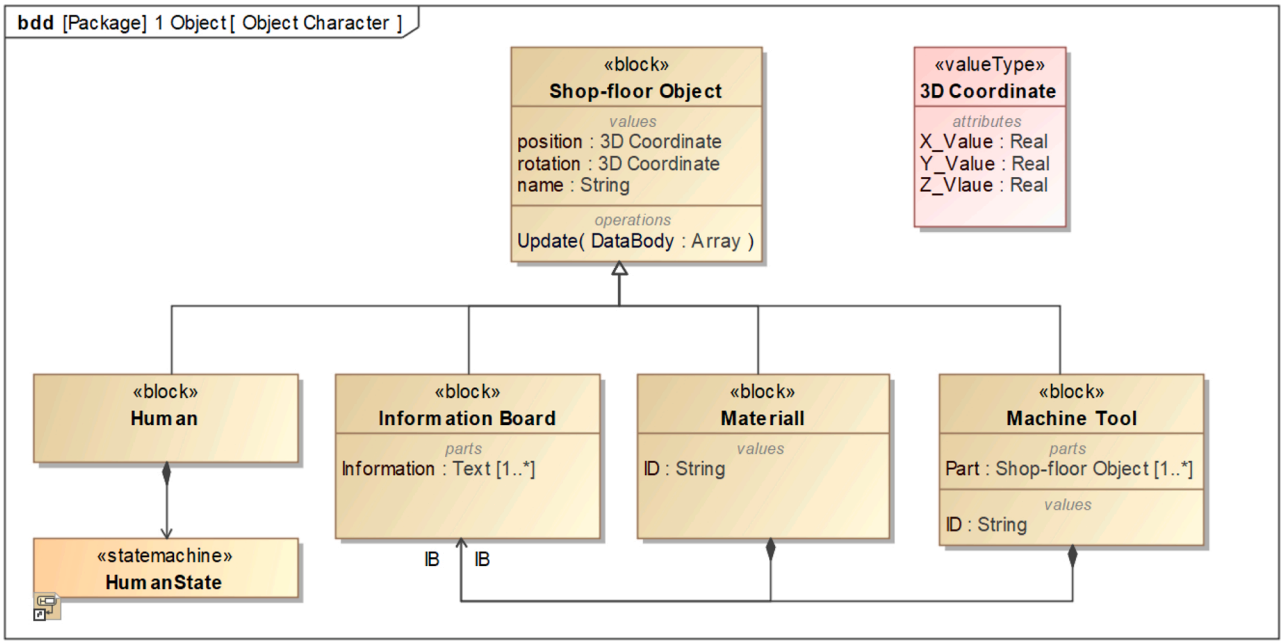

Fig. 22. Block Definition Diagram of component level objects and their attributes.

Human Synchronization Component, i.e., *Human Sync System*. The participant lifelines of the human synchronization interaction are *Database Server*, *Database Reader*, *Database Resolver*, *Human Updater*, and *Human*. *Database Reader* sends a synchronization message called *Execute SQL* to *Database Server* (represented by a solid closed line solid arrow) and calls the *Execute SQL* operation of *Database Server*. After a period of time, *Database Server* returns a reply message (the reply message and its name can generally be omitted, if present, identified by a dotted line open arrow). Upon receipt of the return message, *Database Reader* sends the *Decode* message and parameter values to *Database Resolver*. This message calls the *Decode* operation of *Resolver*. *Resolver* then sends itself a message to call its own *Test Effectiveness* operation. When the condition segment represented by the orange box is executed, if the data validity is true, the *Analysis Head* is executed. At some point during the operation, a *Search* message is sent to *Human Updater*, which takes the *Data Head* as a reference to the *Human ID* parameter and triggers the *Search* operation of *Human Updater*. After completing the *Analysis Head* operation, the *Analysis Body* operation is carried out, and then the *Assign Value* message is sent to the *Human Updater*. Afterwards, the *Update* message is sent to the *Human* object, so the *Human* object can update its position and rotation properties. If the data are invalid, an Error message is returned to the outside world.

In addition to the general behavioral characteristics of the time sequence, the *Human* object in the SDT is also state-based. That means during the operation cycle of the SDT, a *Human* object is in a state with a definite meaning, such as idle, walking, work, and handling, at any moment in the lifecycle. A State Machine Diagram is a behavior diagram that provides a detailed design of a system's behavior, focusing on how objects change state over time. The State Machine Diagram is applied to further model the behavior characteristics of the Human, as shown in Fig. 26. The data source of the *Human* status is the IoT, and most of the data are position information. The position-based state is only static or

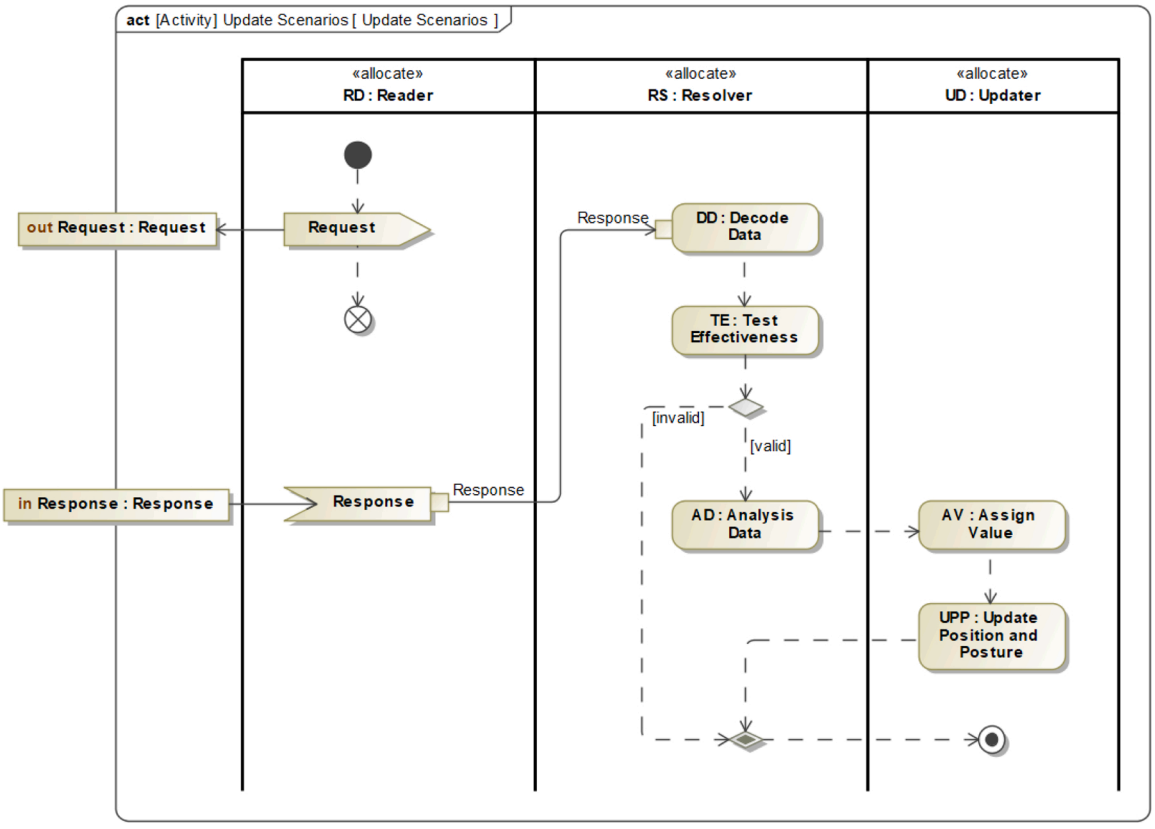

Fig. 23. Action assignment for *Update Scenarios* activities.

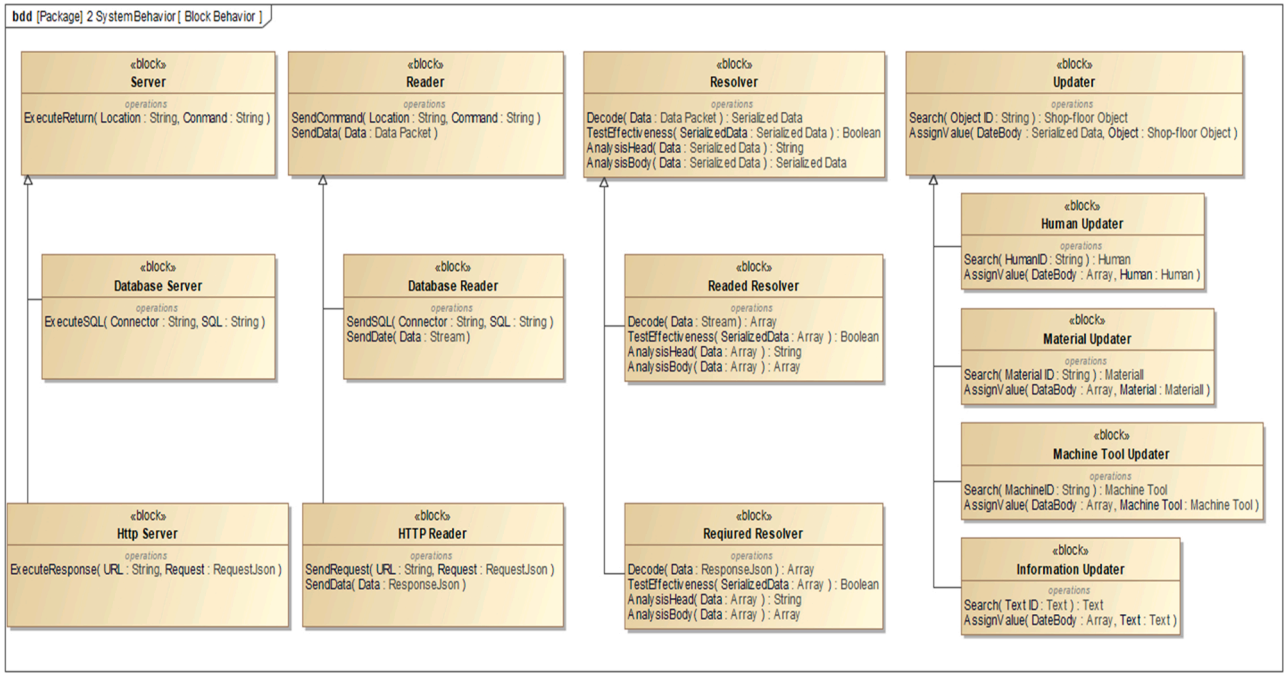

Fig. 24. Block Definition Diagram of subsystem-level behavior for the synchronization system.

moving, so the *Human* status is divided into two basic states: *Static* and *Move*. The initial state node is identified by solid dots (●), and the state connected to the initial node is the default state. The *Static* state is a composite state whose default state is *Idle*, which changes to *Operate* state when an *Operate* message is received. When a *Stop* message is received, the *Idle* state is returned. Whether idle or operate, as soon as a *Move* message is received, it changes to a *Move* state. The motion state is a composite state, which defaults to *Walk*. When a *Carry* message is received, the *Walk* state changes to *Carry* and *Walk*. Whether on foot or in handling, as soon as a *Stop* message is received, it changes to the *Static* state.

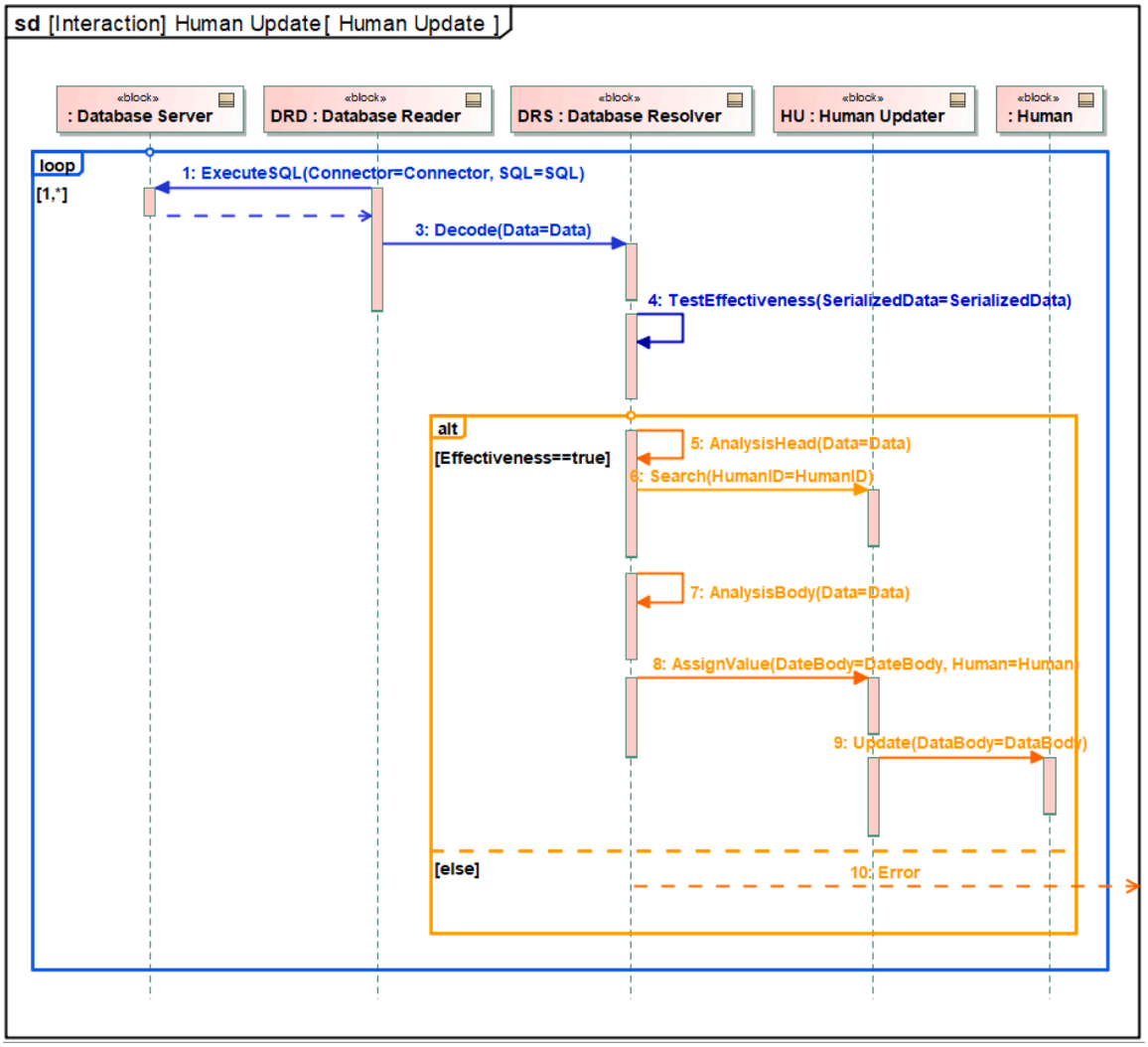

Fig. 25. Sequence Diagram of component interaction for Human Sync System.

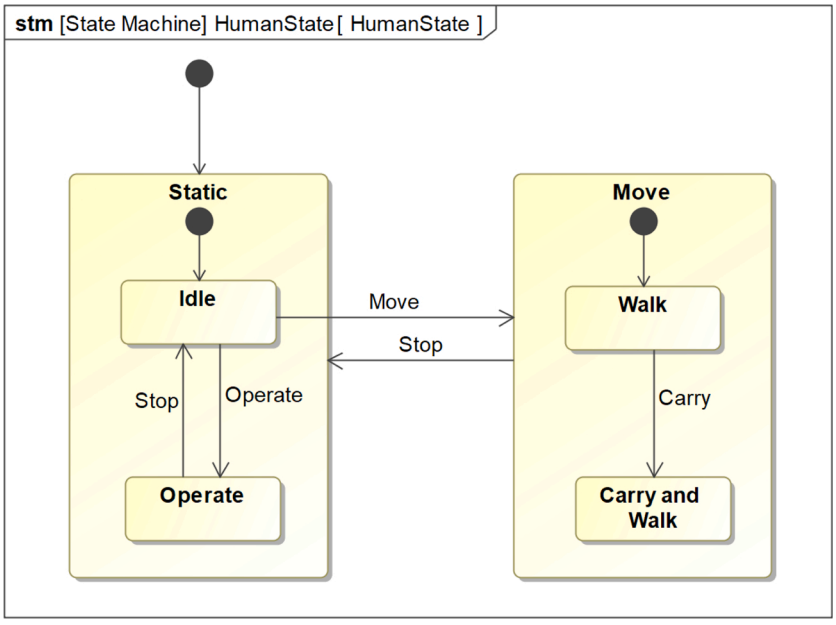

Fig. 26. State Machine Diagram of Human Status.

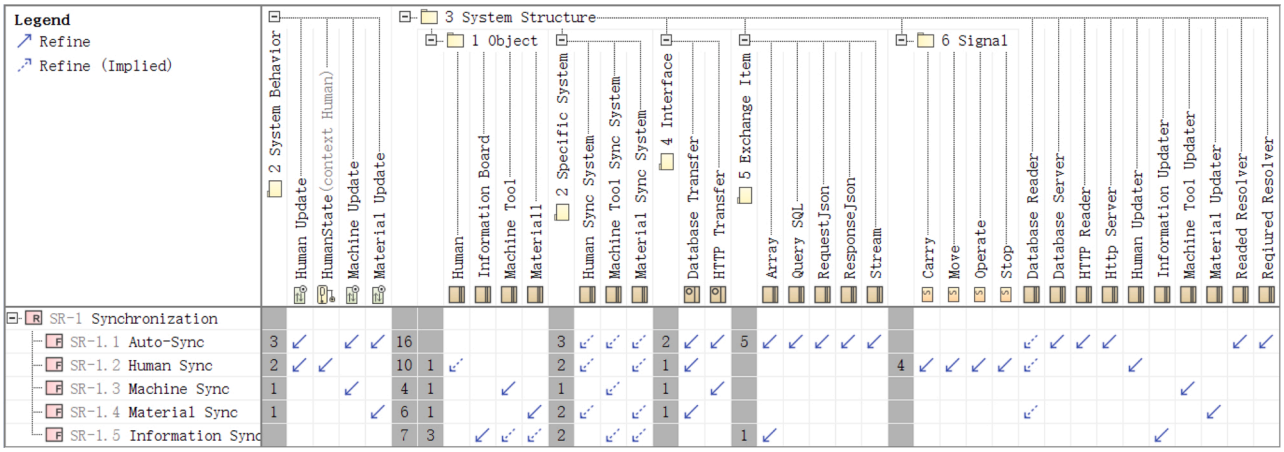

Fig. 27. Traceable refined matrix of the model elements to the requirement for the synchronization system.

5.4. SS1: Phase 2 of the synchronization system requirements

Fig. 27 shows a retrospective relationship between the system model elements and system requirements for the synchronized system.

6. Implementation domain

6.1. Realization of software development of subsystems

The problem domain and solution domain models only clarify the relationships among the requirements, structures, behaviors, and parameters at the system, subsystem, and component levels, including the objects and their interrelationships, processes and their timing relationships, and object-process assignment relationships. However, how model elements are implemented and what technologies to use are not considered. The implementation domain realizes and instantiates the object elements including the block, signal, and value type, and to writes the logic structure of the process elements, such as activity, operation, and reception. Model elements that are not universal in the system model are specialized and realized based on technical details. Each block of the system involves different disciplines and technologies, so the implementation domain of a system is often a stage in which multidisciplinary teams work together. This section realizes the visualization system, synchronization system, and simulation system in the implementation domain.

6.1.1. Realization of the visualization system

First, the shop floor is surveyed offline and the shop-floor layout is drawn using AutoCAD. Second, we use Creo and Solidworks to model the product. We obtained the equipment model from the equipment manufacturer. Used Creo, Solidworks, UG, and SketchUp for the model analysis and the preliminary lightweight, and 3DS MAX for the format conversion and further lightweight. Afterwards, we used Rhino to model non-product and non-equipment objects on the shop floor, and used Mixamo Fuse to model humans. Finally, we used Unity3D, a real-time 3D rendering engine, as a layout and rendering tool. It is a game development engine that can satisfy the visualization requirement of the SDT. The modeling, layout, and rendering of an SDT are shown in Fig. 28.

6.1.2. Realization of the synchronization system

The Human Sync System is discussed here as an example, and the technical scheme is shown in Fig. 29.

(1) **Model ontology**: We created a 3D model of humans. Sixty-six key skeletons that determined the posture of a person are modeled, and the skeleton is abstracted as a connecting rod to establish a human skeletal model. The human model includes the head, neck, chest, shoulder, upper arm, lower arm, wrist, finger joints, abdomen, waist, pelvis, thigh, lower legs, feet, and toe joints. Afterwards, the muscles are created, and the appearance structure is constructed outside the muscles. Finally, the appearance pictures are added. In the stage of static modeling, the 3D model of humans is constructed. During the operation of the system, only the position, rotation and action attributes of humans are updated; the model ontology is not updated.
(2) **Motor function:** We adjust and bind the articulation relationship of the human skeletons, and restrain the rotation direction and rotation angle range of each joint to prevent abnormal postures.
(3) **State machine method:** We use a state machine to control the humans' behavior. The realization of the state machine can be divided into the state machine design and state transfer algorithm.

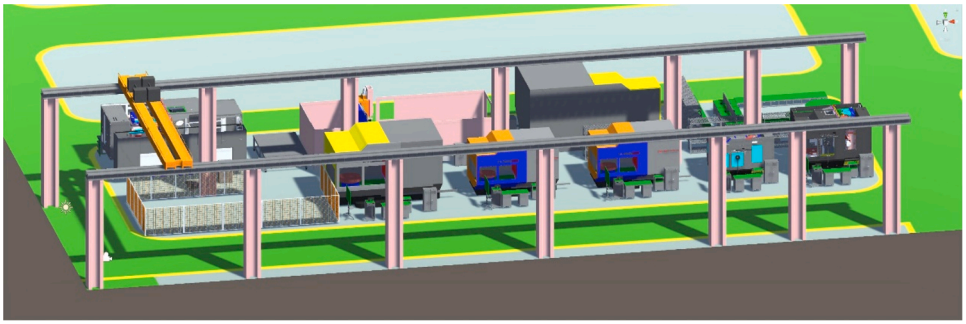

Fig. 28. Modeling, layout and rendering of an SDT.

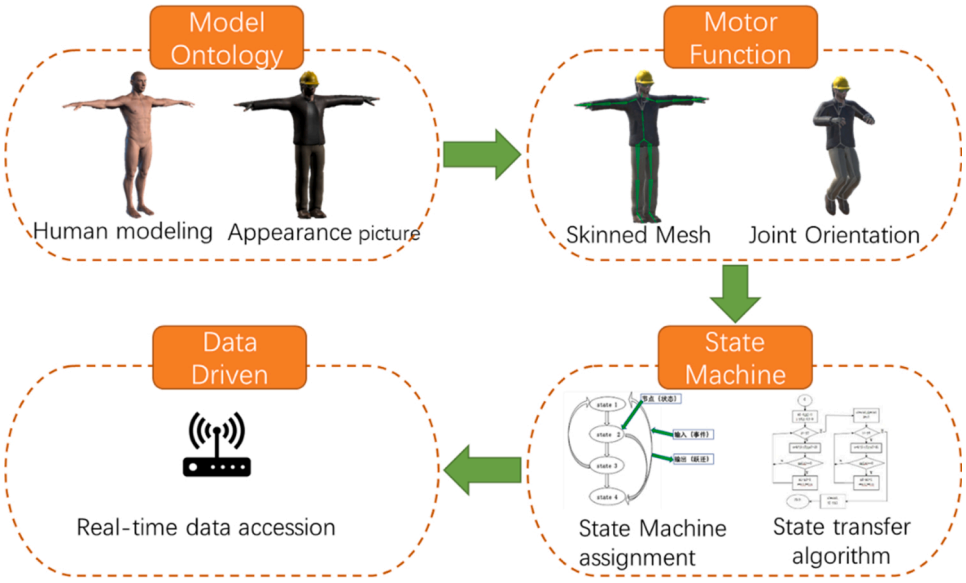

Fig. 29. Human synchronization technology scheme.

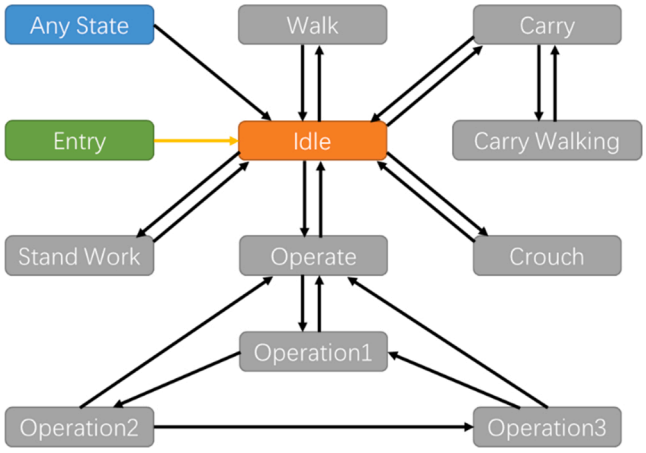

Fig. 30. Refinement of the human state machine.

(4) **Data-driven:** We use real-time data to drive the state machine to run according to the state transfer algorithm, thereby driving the synchronous operation of the humans in the model.

The detailed state machine of humans is shown in Fig. 30. The default state of a person is "idle". The "idle" state and the "walking", "lifting", "squatting", "operation", and "standing work" state can be directly converted into each other, and the "lifting" state and the "handling" state can be converted into each other. "Operation" is an abstract state that is implemented as three operation subclasses. To show the operation state of a person in action, three general operation states of one-way loops are used as agents. Any of these operation states can be restored to the abstract state of "operation." Besides, any state can be converted to "idle" state.

The Human status transfer algorithm is shown in Fig. 31. Real-time location data of materials and personnel are collected by UWB equipment and stored in the UWB database of shop-floor IoT system. The real-time status data of materials and personnel are collected and stored in MES. The synchronization system obtains the real-time data from the IoT system and MES through data integration interfaces to drive the movement of the model in the virtual shop floor. Hence, the location is updated based on the UWB data, and the status is updated according to the MES data. The specific operation steps are as follows. According to the state machine method, different operation actions such as walking, carrying, standing, and sitting are added to the human ontology model, trigger conditions are preset, and it is bound as a human prefab in Unity3D. In the UWB database, ID is used as the unique primary key of the human, and the coordinates, status and other data corresponding to the character can be queried in the database through the ID. When the SDT obtains the real-time data of a human with new ID in the UWB database, it instantiates the human prefab at the specified coordinate position according to the data, and adds the specified action to it.

The variable i represents the update times and t represents the local clock. For the i th update, set the clock t to 0, read the latest position P_i of the label carried by the person from the UWB database, and calculate the distance $\left|\overrightarrow{P_i} - \overrightarrow{P}\right|$ between the current position of the person object P and P_i. If the distance is less than the set value L, the person is considered to be stationary; otherwise, the person is considered to be in motion. For the stationary state, the existence of an operation signal is further judged according to the technical information. If it exists, change the person's state to "operation"; otherwise, set the person's state to "idle". Then assign the current position of the person to the latest position P_i. For the motion state, further judge whether there is a handling signal according to the information of handling task and logistics. If there is, change the state of the person to "handling". Otherwise, change the state to "walking". Whether "handling" or "walking", the motion drives the person move to the target position P_i. After one update, accumulate the clock t. When the accumulated value is greater than the set value T, reset the clock to 0 and update the number $i = i + 1$ for the next update.

To realize real-time synchronization of the virtual and physical shop floor, the IoT-based data communication scheme is established, as shown in Fig. 32. Create the SQL configuration file, extract the database connection information from the configuration file, and create the connection when the SDT starts to run. Then, we send the SQL instructions to the RFID, UWB, and MES databases, and store the returned results in the data sheet. Finally, we extract the query results to realize real-time data communication. It should be noted that as the location data of humans comes from the IoT system that uses the MySQL database in this case, the same type of database is used in the synchronization system to minimize unnecessary workload during the integration process.

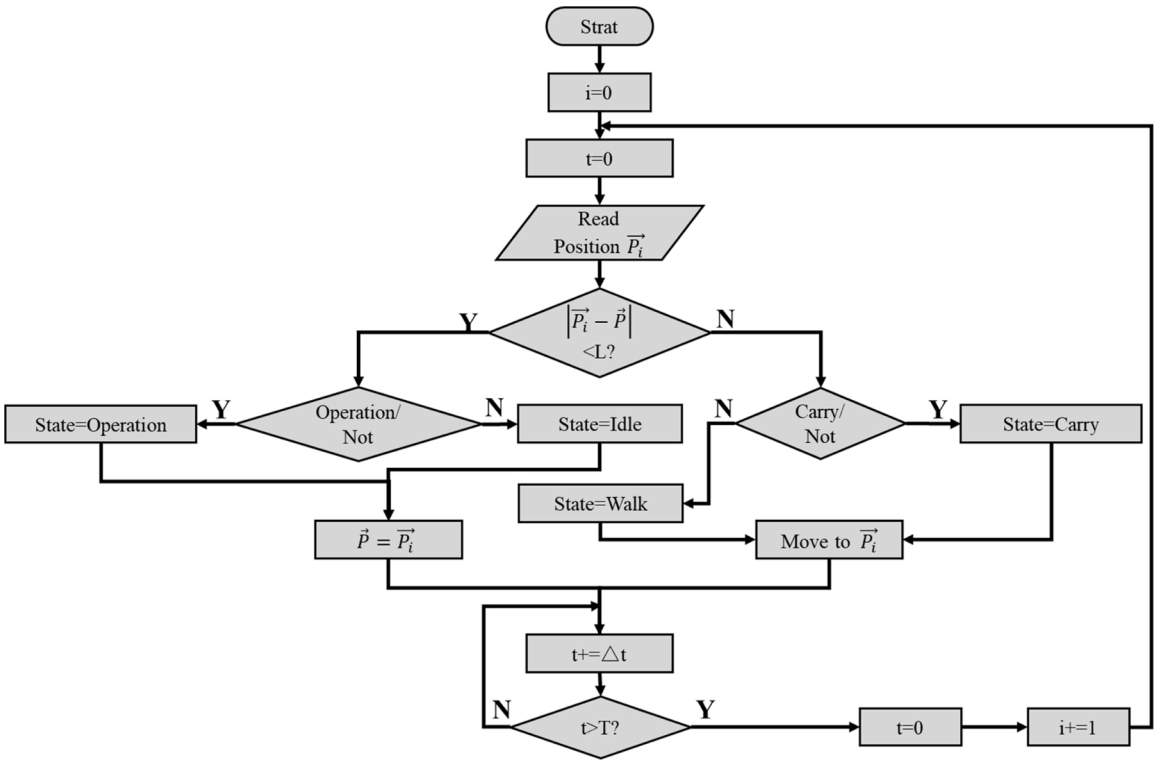

Fig. 31. Human state transfer algorithm.

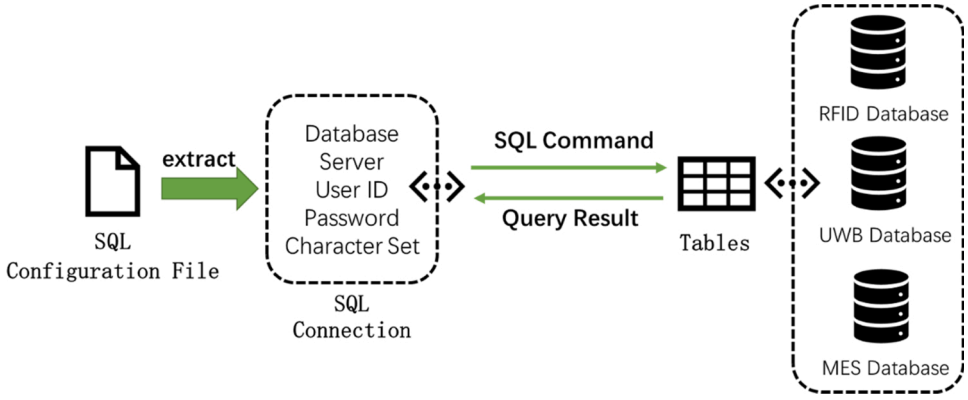

Fig. 32. IoT-based data communication scheme.

6.1.3. Realization of the simulation system

To realize continuous transient simulation, the event-driven scheduling method is adopted. The event-driven scheduling method is a simulation propulsion mechanism based on the Future Event List (FEL) to ensure that all events occur in the correct sequence. In this method, except for the initial event, other events are not pre-arranged, but occur naturally, such as random arrival and random end.

First, the system input characteristics of the simulation object should be modeled. In this case, the operation logic of shop floor is analyzed and the actual production processes, such as arrival event, processing end event, and departure event, are described with abstract models. Second, according to the historical shop-floor data, the distribution types of different events are identified and the distribution parameters estimated. Afterwards, the generated simulation samples are screened according to simulation requirements and the actual shop-floor's conditions in order to determine the sample input generator. Finally, the processing logic of different simulation events is constructed based on the production process, and the processing logic for different events is triggered by the advancement of the simulation clock in order to realize the simulation process.

When the simulation clock Clock $= t$ is approaching time t_1 ($t \rightarrow t_1^-$), the system snapshot is as shown in Table 1, including the system status and FEL. The current system is in state S_1 and there are n pending events in the FEL. The FEL contains a description of all future events at and after moment t_1, as well as their occurrence time. Assume that only two types of events occur in the system, type A and type B. Then assume that at time t_1, there are n events in FEL: (A, t_1), (B, t_2), (B, t_3) …… (A, t_n). Events in the FEL are sequenced by time, i.e., the event occurrence time satisfies the following:

$$t \le t_1 \le t_2 \le t_3 \le \ldots \le t_n \tag{1}$$

When the system snapshot of the simulation clock Clock $= t$ is updated, the simulation system proceeds as follows:

Step 1: Remove from FEL the events that occur at time t_1, that is (A, t_1);

Step 2: Push Clock to t_1, that is $t = t_1$;

Table 1
System snapshot at time t.

Clock	System state	FEL
$t \rightarrow t_1^-$	(S_1, n)— —System state is S_1 at time t_1, and there are n pending events in the FEL	(A, t_1)— —Type A event occurs at t_1 (B, t_2)— —Type B event occurs at t_2 (B, t_3)— —Type B event occurs at t_3 … … (A, t_n)— —Type B event occurs at t_n

Step 3: Execute event A (Update system status, change entity attributes, set variable values as needed);

Step 4*: If needed, generate future events and place them in FEL in chronological order;

Step 5: Update statistical counter.

After Step 3, the system is in state S_2. After Step 4 is executed, new events are added to the FEL. It is assumed that two future events, $(\mathrm{A}, t_\varsigma)$ and (B, t_η), occurred during the handling of this (A, t_1) event. Assume that the event times t_ς and t_η satisfy

$$t_2 \le t_\varsigma \le t_3, t_3 \le t_\eta \le t_4 \tag{2}$$

Then, assume that at time t_1, there are $n+1$ events in FEL: (B, t_2), $(\mathrm{A}, t_\varsigma)$, (B, t_3), (B, t_η) …… (A, t_n). System snapshots at t_1 are shown in Table 2.

The running process of this method consists of four steps: initialize simulation variables, initialize FEL, start execution policy, and end of simulation. However, the above process is offline and entirely based on historical data. For this reason, a continuous transient simulation based on real-time data is proposed, as shown in Fig. 33.

In the process of real-time data-driven operation of the SDT, the current actual situation of the physical shop floor is endowed with the SDT. Transient simulation can be realized, when the state and actual statistics in the SDT are taken as the initial simulation value and then the simulation is performed. Second, a continuous simulation can be realized when the simulation process is executed in a certain frequency cycle. The orange lines in the figure describe the core of a continuous transient simulation scheme based on real-time data. The details are as follows.

After the FEL is initialized in the simulation model, scan the current process of each material, the current state of each equipment, and the current information of each queue from SDT. Then convert it into the initial event list that is inserted into the FEL. In this case, the conversion rule is that if a part is in the processing station or inspection station, the end of processing event (F event) or departure event (D event) is inserted into the FEL and the status of the corresponding processing station or inspection station is set to "busy." If a part is in the processing or inspection cache area, the part type is counted and added to the corresponding entity queue (Queue).

Table 2
System snapshot at time t_1.

Clock	System state	FEL
t_1	$(S_2, n+1)$— —System state is S_2 at time t_2, and there are $n+1$ pending events in the FEL	(B, t_2)— —Type B event occurs at t_2 (A, t_ς)— —Type A event occurs at t_ς (B, t_3)— —Type B event occurs at t_3 (B, t_η)— —Type B event occurs at t_η … … (A, t_n) — —Type A event occurs at t_n

6.2. Realization of system integration and verification

In this section, the modeling of the system enters the right half of the V model, namely, subsystem verification and integration, system verification, and confirmation. The SDT system is implemented based on an NC machining shop floor that produces spacecraft structures with high speed five-axis machining units and inspection units.

6.2.1. System integration

The following is the process that we use for system integration. First, the shop floor is surveyed offline and the shop-floor layout is drawn using AutoCAD. Second, we use Creo and Solidworks to model the product. We obtain the equipment model from the equipment manufacturer. We use Creo, Solidworks, UG, and SketchUp for the model analysis and the preliminary lightweight, and 3DS MAX for the format conversion and further lightweight. Afterwards, we use Rhino to model non-product and non-equipment objects on the shop floor, and use Mixamo Fuse to model humans. Finally, these visual models are imported into Unity3D, and are integrated with the class library generated by the SysML model by establishing a data communication mechanism between SysML and Unity 3D. In addition, a simulation engine to achieve system integration is developed independently to achieve the integration of all subsystems. It should be noted that to the reproducible environment for constructing an SDT can be realized by using MBSE language and tools during the development of top-level systems to package many reusable model objects such as protocol stacks, network modules, buses, sample generators, and database read-write modules into a standard library. In the subsequent system design and software development, these models can be reused by dragging and dropping according to the actual shop-floor requirements.

The specific implementation process for the construction of an SDT is shown in Fig. 34. First, it is necessary to analyze the requirements, behavior, structure and parameters of SDT, and complete the system design of the problem domain model, solution domain model, and MoEs of the system. Then, the communication mechanism between SysML and Unity 3D is established to realize system integration. The specific implementation steps are as follows: 1) Use the Java coding function embedded in SysML modeling software to establish the communication relationship with the intermediate database; 2) Store SysML model data in the intermediate database through file reading and writing, database, TCP and UDP and 3) establish communication between Unity 3D and the intermediate database, then read the data stored in the intermediate database of SysML, and run the Unity program driven by the SysML model.

The SDT case in this paper has three subsystems: a visualization system, a synchronization system and a simulation system. The specific implementation process of developing these three subsystems in Unity3D are as follows. 1) Based on actual measured geometric data, use commercial modeling software such as SoildWords/UG to model the virtual shop floor, convert the model into FBX format by 3DMAX software, and then import it into the Unity3D platform. In Unity3D, according to the actual shop-floor production layout, add lighting, color and other rendering effects for the virtual shop floor. 2) Create human/product/equipment/UI Kanban model based on Activity Diagram. Add Motor function to the human model ontology, and construct the human state machine to control its movement behavior. For the equipment model, reconstruct the model ontology and define the parent-child relationship between the components, and regulate the origin and movement range of each part. 3) Integrate the SDT system and the IoT system/MES system. The specific implementation process is as follows. First, we write the database connection code into the running script of the SDT system and set the database access method to read the EXCEL configuration file in the script. Secondly, we set the database name, server name, user name, and password in the configuration file that can

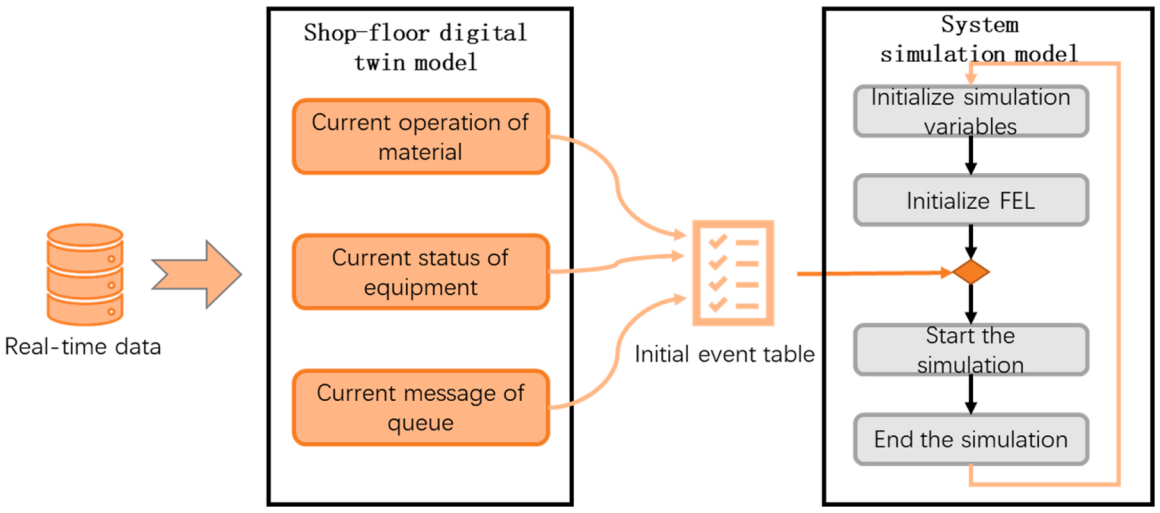

Fig. 33. Continuous transient simulation scheme based on real-time data.

be modifiable. Finally, the SDT system establishes a local area network (LAN) with the IoT system, and remotely accesses the UWB, RFID, and MES databases by modifying the relevant settings in the configuration file. 4) Scan the real-time status of the SDT, cyclically update the FEL, and complete the online simulation of the shop-floor operation status based on the input characteristic model, sample generator and event processing logic constructed in SysML. Finally output the simulation results on the Kanban of the visualization system.

Fig. 35 shows a full picture of the SDT system in normal operation.

6.2.2. Verification of the system function requirements

The synchronization of equipment actions, equipment parameters, human, and real-time simulation of the SDT correspond to the stakeholder needs numbered SN-7.1, SN-7.2, and SN-7.3, respectively, in Fig. 4.

(1) Verification of the machine tool synchronization function

The data related to machine tools on the shop floor comes from the machine tool information collection system, with which the SDT system receives machine tool data in real time through Redis integrated communication. Fig. 36 shows the synchronization picture of the machining action of No. 6 machine tool MT-24/32-3S in this test case. The moving axes of the machine tools in the system have different colors. When the operator with the UWB tag starts the machine and the workpiece with the RFID tag enters the illustrated workstation, the machine moves synchronously according to the actual action of the machine tool in the system. Fig. 37 shows the synchronization picture of information Kanban of No. 1 Machine Tool DMU80FD. After the operator starts the machine, the information Kanban of the machine tool in the system changes from null to having data. When the operator changes the spindle parameters, the relevant values in the system appear.

(2) Verification of the human synchronization function

Fig. 38 shows the synchronization of the humans in the test case. According to the method described in Section 6.1.2, the 3D models of the humans are constructed and are first given the motor function. Then a state machine that controls the humans' behavior is built and the actions such as standing, walking, carrying, operating and sitting were preset for the humans. Finally, the real-time position data of the humans in the UWB database are obtained in real time to drive the human model to change its state and move. For example, when an operator with UWB tags starts moving from a standstill with an RFID tagged workpiece, the virtual person in the SDT changes from the "Idle" state to the "Carry and Walk" state, and from a standing position to a handling position, capturing the object of the workpiece and moving.

(3) Verification of the material synchronization function

The process is as follows. First, the current tag-carrying materials are read from the RFID and UWB database. The corresponding model is selected from the model library based on the model information and instantiated in the SDT system. Secondly, the detailed information of the material are queried from MES and displayed in the information Kanban of material. Then, the real-time data are obtained from RFID, UWB, and MES databases, and preprocess these data are preprocesed. Finally, based on real-time data, the material's location information and processing information are updated, which is manifested as the capture of fixtures and workpieces by the station, and the capture of workpieces by the transport personnel. Fig. 39 shows the synchronization of the material. The human places the workpiece with the RFID tag on the desk in front of the machine tool. Once the RFID reader deployed on the desk senses the workpiece tag, the workpiece in the system immediately becomes a sub-object of the desk. A mouse is used to click on the workpiece to display its information Kanban, where the type, quantity, and specification of the workpiece are consistent with the input information from the shop-floor IoT.

(4) Verification of the simulation function

The total simulation time of the simulation system is set at 500 h and the simulation frequency is 1 time/minute. Fig. 40 shows the system simulation Kanban under this condition. The simulation report data are evaluated by experts and has a certain reference value. The simulation report is updated one minute later.

6.2.3. Validation of the system indicator requirements

Stakeholder needs numbered SN-7.2.1 and SN-7.2.2 specify two indicators of model completeness and synchronous timeliness respectively.

The test results of model completeness in this case are shown as Formula 3.

$$\text{Model completeness} = \frac{\text{Number of virtual shop} - \text{floor actual models}}{\text{Number of physical shop} - \text{floor actual objects}} \times 100\% = \frac{102}{106} \times 100\% = 96.22\% \tag{3}$$

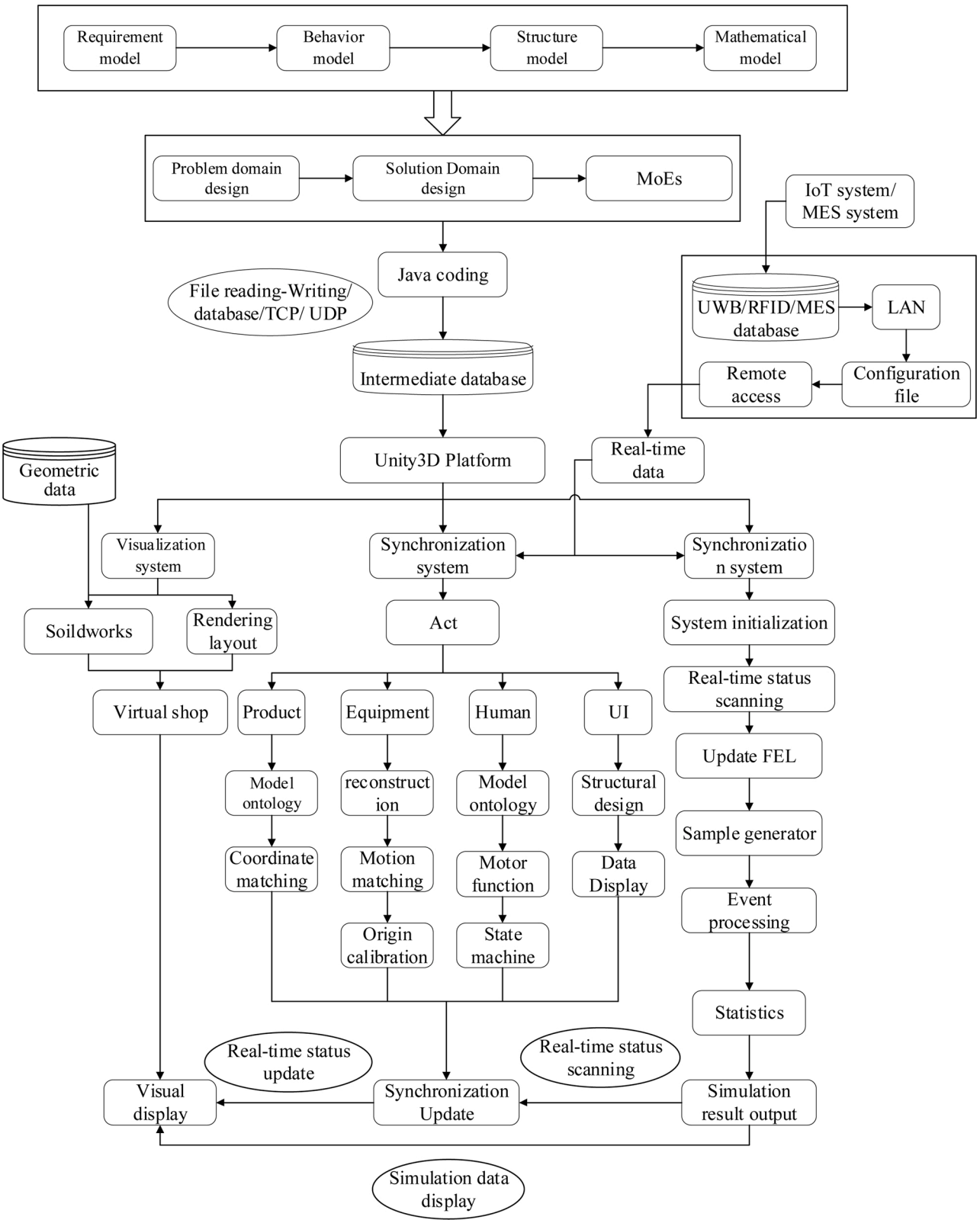

Fig. 34. Implementation process of constructing an SDT using MBSE.

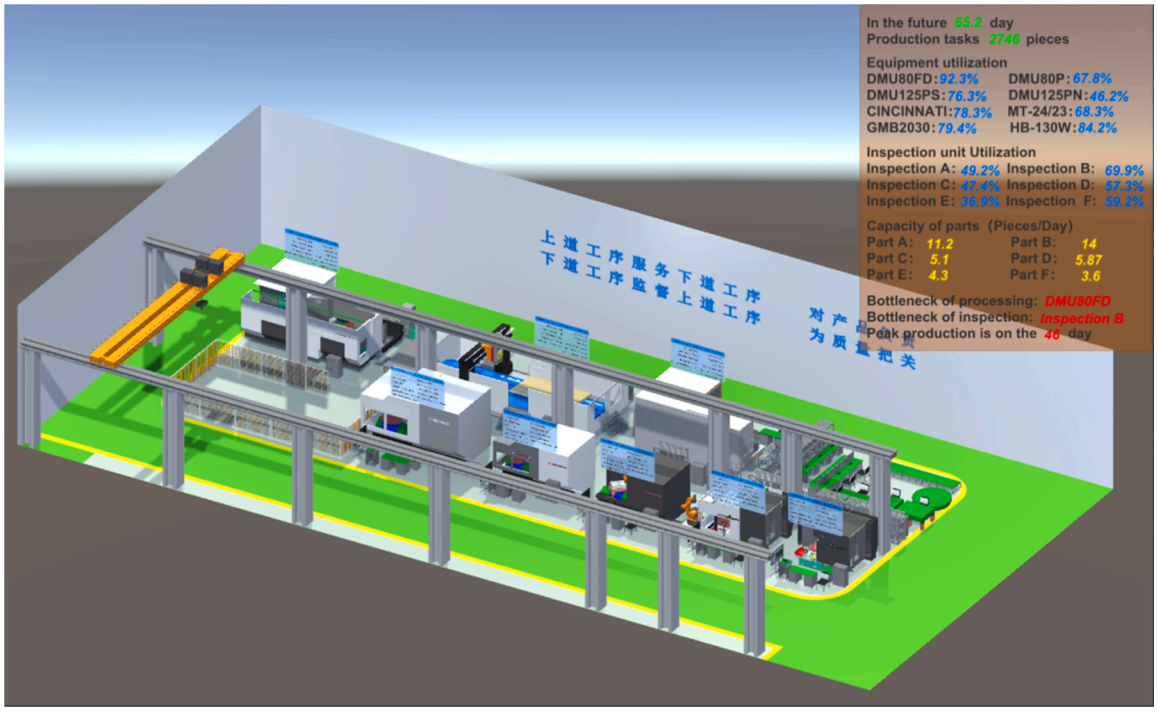

Fig. 35. Full picture of the SDT system in normal operation.

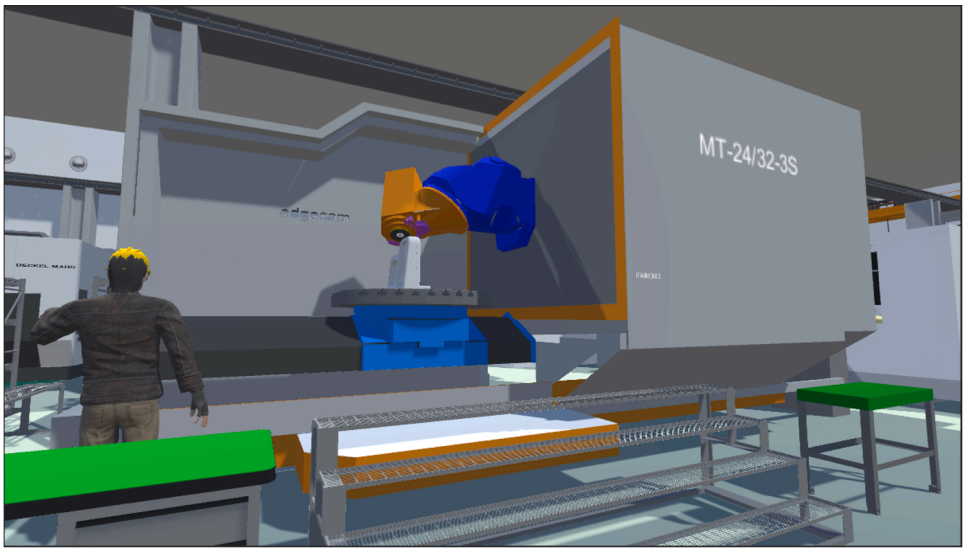

Fig. 36. Synchronization of the machine tool actions.

In order to ensure the real-time and visual monitoring of the SDT while reducing operating costs, the burden on servers, and network requirements, the allowable maximum delay is 3 s. Using a stopwatch, the synchronous timeliness test is completed by the way of workpiece in and out of the station. The time difference between the time when the workpiece enters and leaves the station is recorded in the system and the time in the actual shop floor. The test is repeated five times and the average time delay calculated. The test results in this case are shown in Table 3. When the update cycle is less than 2.5 s, the maximum delay can be controlled within 3 s.

7. Conclusion

How to build an SDT model is the prerequisite for applying DT technology in the manufacturing phase, which is also a hotspot for scholars and companies. In view of the current shop-floor digital twin construction process, focusing on theory but neglecting demand, low reuse rate of system function modules, and lack of scalability, this paper proposes an MBSE-based construction method of SDT. This method is oriented to the application needs of the shop floor, and clearly describes the composition of the production elements of the SDT and their relationship, the composition of the system function modules and their solutions, and the external integration interfaces through the digital model, making the software development process of the SDT manageable, reproducible, reusable, traceable and extensible. The main contributions of this paper are as follows:

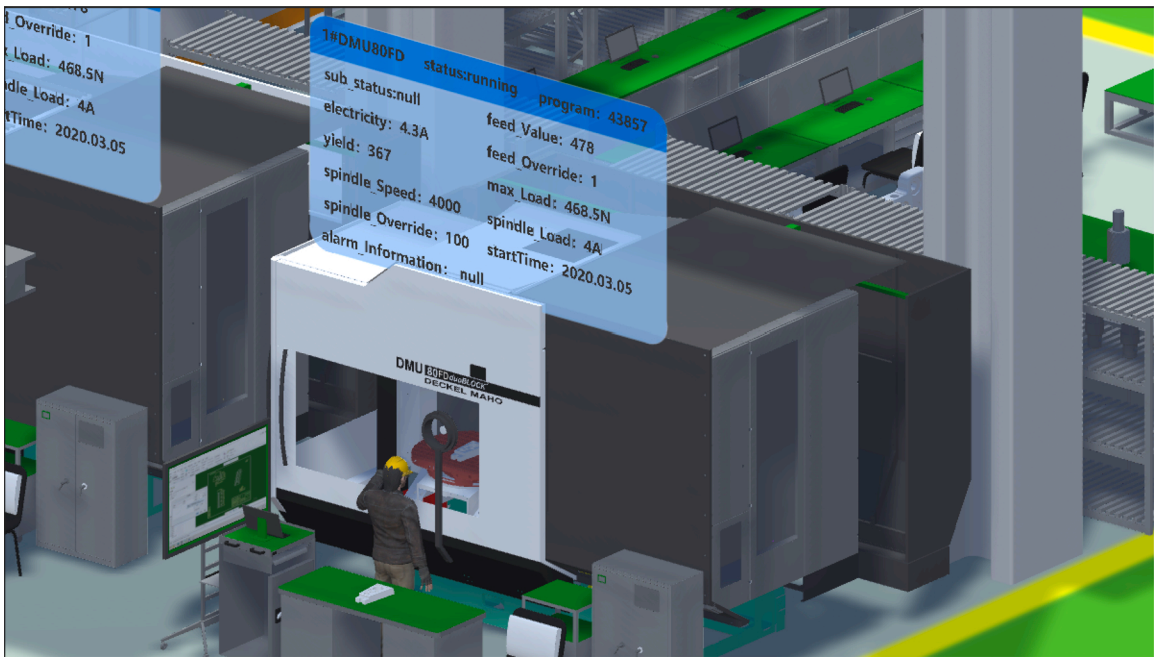

Fig. 37. Synchronization of the information Kanban of the machine tool.

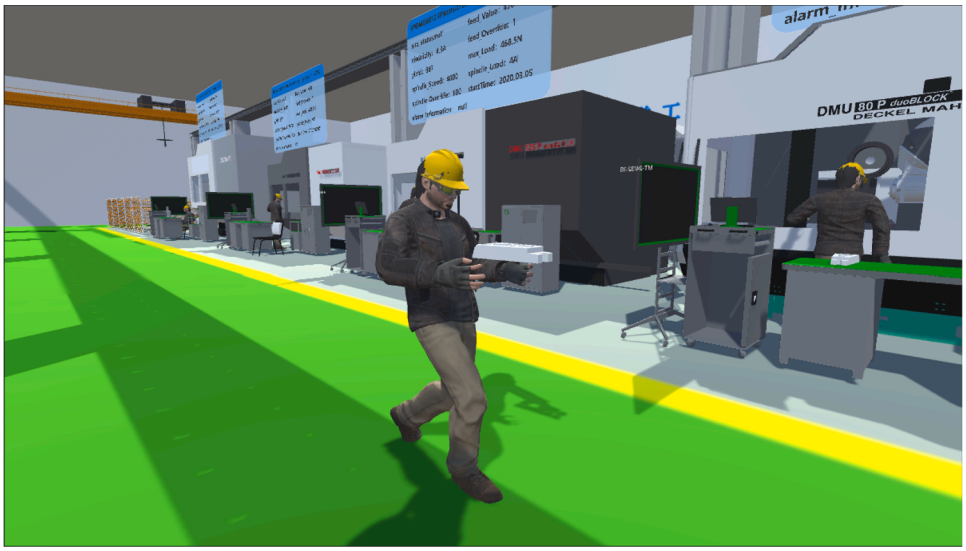

Fig. 38. Synchronization of a human.

(1) The shop floor is a typical complex manufacturing system, with the characteristics of large scale, complex structure, and interrelated coupling among various components. There are a great deal of information transmission, resource coordination and production disturbance during the operation process of the shop floor, and the evolution of the shop-floor operation is nonlinear and dynamic. In the process of development for SDT, it is very easy to make omissions and errors. The MBSE-based modular development method proposed in this paper is based on requirements. It can not only avoid errors in the top-level design stage of the system through clear requirements description and intuitive design model diagrams, but can also form a normalized and standardized model library. When creating a cyber-physical production system model with similar functional modules, we can reuse the modules and in-depth custom design according to the actual conditions of different shop floors to reduce the software development time.

(2) The method proposed in this article improves the format of describing design schemes and data in words or pictures. It is helpful for developers to grasp the overall situation of the shop floor more intuitively and completely, and the entire construction process from requirement analysis to system design to test verification can be traced completely. This greatly reduces the number of problems that may occur in the process of system development and can ensure the practicability of the system. Moreover, this method has established a requirements-based cyber-physical production system R&D framework from "function to meet requirements" to "requirements-driven design," which is traceable and verifiable to the entire construction process. Furthermore, the method is more universal.

Fig. 39. Synchronization of the material and its information Kanban.

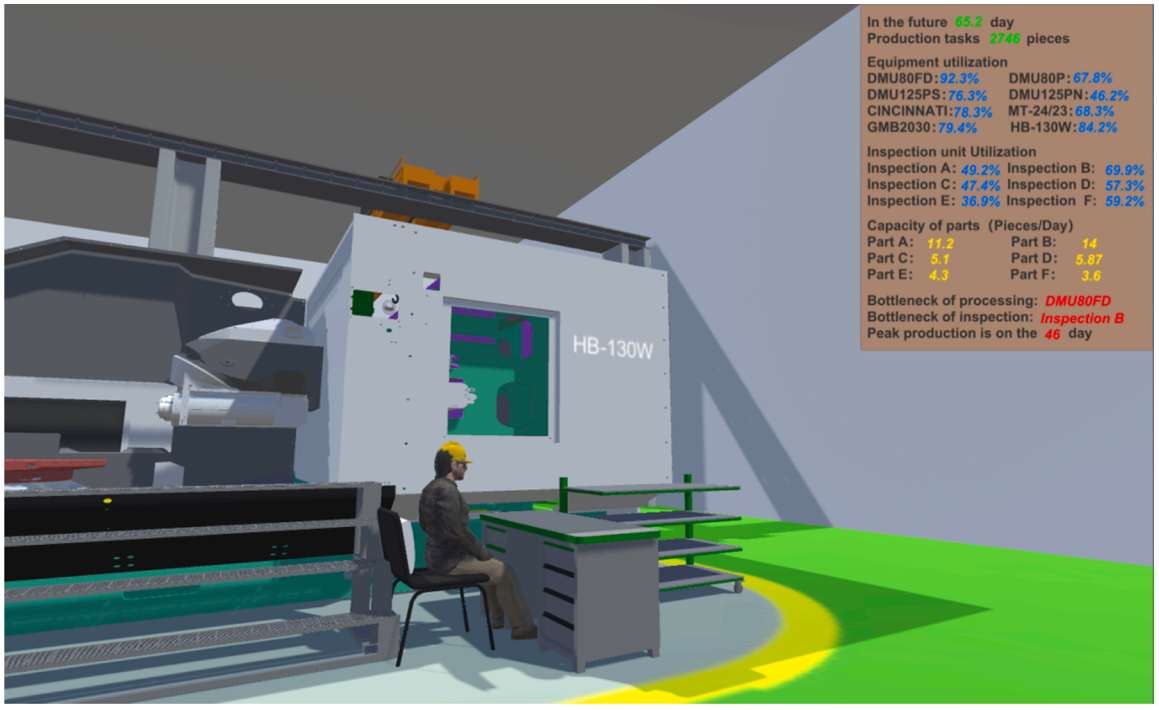

Fig. 40. Kanban of the simulation system.

Table 3
Test results of synchronous timeliness.

Update cycle	Average delay (s)	Maximum delay (s)	Minimum delay (s)	Delay interval	In 3 s (Y/N)
1 s	1.971	2.6	1.52	[1.52, 2.6]	Yes
1.5 s	2.508	2.83	2.12	[2.12, 2.83]	Yes
2 s	2.771	2.99	2.51	[2.51, 2.99]	Yes
2.5 s	3.246	3.52	3.03	[3.03, 3.52]	No

(3) A continuous transient simulation method for a discrete manufacturing system based on real-time data is proposed. The proposed method integrates real-time data into the simulation process of a discrete event system, which can realize the real-time simulation by means of state cycle scanning, the sustainability of simulation by means of cycle execution, and the transient simulation based on the event-driven scheduling mode. Compared with the forecasting method based on big data technology, this method combines online real-time state scanning and an offline system simulation. It has strong correlation with the system model, and real-time data is fully considered. This approach also has the characteristic of occurring in real-time, and is sustainable and transient, which can provide real-time dynamic decision guidance for shop floor production.

The MBSE methodology provides a new way of thinking for the construction of SDT and the software development process. However, the cases discussed in this article have some limitations, especially in the simulation function module. In the follow-up, we will discuss how to use MBSE methodology and complex networks to model the complex relationship between multi-dimensional and multi-level production factors and multi-source heterogeneous production data for multi-variety and variable-batch discrete complex manufacturing shop floor, and build a large integration on this basis. The DT prediction model integrating the data analysis model and the mechanism model enables accurate online prediction of the operating status of the shop floor.

Declaration of Competing Interest

The authors report no declarations of interest.

Acknowledgments

The authors would like to express our sincere gratitude to the anonymous reviewers for the invaluable comments that have improved the quality of the paper. We also thank LetPub (www. letpub.com) for its linguistic assistance during the preparation of this manuscript. This research is financially supported in part by the National Key Research and Development Program of China (No. 2020YFB1710300), in part by the National Natural Science Foundation of China (No. 52005042 & 51935003), and in part by the National Defense Basic Scientific Research Program of China (No. JCKY2016204A502).

References

[1] Monostori L, Kádár B, Bauernhansl T, Kondoh S, Kumara, Reinhart G, et al. Cyber-physical systems in manufacturing. CIRP Ann-Manuf Techn 2016;65:621–41.
[2] Zhang L, Luo Y, Tao F, Li B, Ren L, Zhang X, et al. Cloud manufacturing: a new manufacturing paradigm. Enterp Inf Syst 2014;8(2):167–87.
[3] Tao F, Qi Q, Liu A, Kusiak A. Data-driven smart manufacturing. Int J Ind Manuf Syst Eng 2018;48:157–69.
[4] Zhuang C, Gong J, Liu J. Digital twin-based assembly data management and process traceability for complex products. J Manuf Syst 2021;58:118–31.
[5] Grieves M. Digital twin: manufacturing excellence through virtual factory replication. 2014. www.apriso.com/library/Whitepaper_Dr_Grieves_DigitalTwin_ManufacturingExcellence.php.
[6] Grieves M. Virtually perfect: driving innovative and lean products through product lifecycle management. Cocoa Beach, FL: Space Coast Press; 2011.
[7] Tao F, Qi Q. Make more digital twins. Nature 2019;573:490–1.
[8] Park KT, Nam YW, Lee HS, Im SJ, Do Noh S, Son JY, et al. Design and implementation of a digital twin application for a connected micro smart factory. Int J Comput Integr Manuf 2019;32:596–614.
[9] Leng J, Zhang H, Yan D, Liu Q, Chen X, Zhang D. Digital twin-driven manufacturing cyber-physical system for parallel controlling of smart shop-floor. J Ambient Intell Humaniz Comput 2019;10(3):1155–66.
[10] Lu Y, Liu C, Kevin I, Wang K, Huang H, Xu X. Digital Twin-driven smart manufacturing: connotation, reference model, applications and research issues. Robot Comput Integr Manuf 2020;61:101837.
[11] Modoni GE, Caldarola EG, Sacco M, Terkaj W. Synchronizing physical and digital factory: benefits and technical challenges. Procedia Cirp 2019;79:472–7.
[12] Liu M, Fang S, Dong H, Xu C. Review of digital twin about concepts, technologies, and industrial applications. J Manuf Syst 2021;58:346–61. https://doi.org/10.1016/j.jmsy.2020.06.017.
[13] Tao F, Zhang H, Liu A, Nee AYC. Digital twin in industry: state-of-the-art. IEEE T Ind Inform 2019;15(4):2405–15.
[14] Tao F, Zhang M. Digital twin shop-floor: a new shop-floor paradigm towards smart manufacturing. IEEE Access 2017;5:20418–27.
[15] Sun X, Bao J, Li J, Zhang Y, Liu S, Zhou B. A digital twin-driven approach for the assembly-commissioning of high precision products. Robot Comput Integr Manuf 2020;61:101839.
[16] Leng J, Liu Q, Ye S, Jing J, Wang Y, Zhang C, et al. Digital twin-driven rapid reconfiguration of the automated manufacturing system via an open architecture model. Robot Comput Integr Manuf 2020;63:101895.
[17] Liu C, Jiang P, Jiang W. Web-based digital twin modeling and remote control of cyber-physical production systems. Robot Comput Integr Manuf 2020;64:101956.
[18] Söderberg R, Wärmefjord K, Carlson JS, Lindkvist L. Toward a Digital Twin for real-time geometry assurance in individualized production. CIRP Ann-Manuf Tech 2017;66(1):137–40.
[19] Kong T, Hu T, Zhou T, Ye Y. Data construction method for the applications of shop-floor digital twin system. J Manuf Syst 2021;58:323–8. https://doi.org/10.1016/j.jmsy.2020.02.003.
[20] Xia K, Sacco C, Kirkpatrick M, Saidy C, Nguyen L, Kircaliali A, et al. A digital twin to train deep reinforcement learning agent for smart manufacturing plants: environment, interfaces and intelligence. J Manuf Syst 2021;58:210–30. https://doi.org/10.1016/j.jmsy.2020.06.012.
[21] Zhuang C, Liu J, Xiong H. Digital twin-based smart production management and control framework for the complex product assembly shop-floor. Int J Adv Manuf Technol 2018;96:1149–63.
[22] Tao F, Cheng Y, Cheng J, Zhang M, Qi Q. Theory and technologies for cyber-physical fusion in digital twin shop-floor. Comput Integr Manuf Syst 2017;23(8):1603–11.
[23] Zhuang C, Miao T, Liu J, Xiong H. The connotation of digital twin, and the construction and application method of shop-floor digital twin. Robot Comput Integr Manuf 2021;68:102075.
[24] Zhao H, Liu J, Xiong H, Zhuang C, Miao T, Liu J, et al. 3D visualization real-time monitoring method for digital twin workshop. Comput Integr Manuf Syst 2019;25(6):1432–43. https://doi.org/10.13196/j.cims.2019.06.011.
[25] Wenzel S, Jessen U. The integration of 3D visualization into the simulation-based planning process of logistics systems. Simulation 2001;77(3–4):114–27.
[26] Do TV. A new solution for a queueing model of a manufacturing cell with negative customers under a rotation rule. Perform Eval 2011;68(4):330–7.
[27] Li Z, Liu G, Hanisch HM, Zhou M. Deadlock prevention based on structure reuse of Petri net supervisors for flexible manufacturing systems. IEEE Trans Syst Man Cybern A Syst Hum 2012;42(1):178–91.
[28] Ning R, Liu J, Tang C. Modeling and simulation technology in digital manufacturing. Chinese J Mech Eng 2006;42(7):132–7.
[29] Zhang Y, Wang W, Wu N, Qian C. IoT-enabled real-time production performance analysis and exception diagnosis model. IEEE Trans Autom Sci Eng 2016;13(3):1318–32.
[30] Schleich B, Anwer N, Mathieu L, Wartzack S. Shaping the digital twin for design and production engineering. CIRP Ann Manuf Technol 2017;66(1):141–4.
[31] Boschert S, Rosen R. Digital twin—the simulation aspect. Mechatronic futures. Berlin, Germany: Springer, Cham; 2016. p. 59–74.
[32] Cheng J, Zhang H, Tao F, Juang CF. DT-II: digital twin enhanced Industrial Internet reference framework towards smart manufacturing. Robot Comput Integr Manuf 2020;62:101881.
[33] Siemems. Digital Twin. Available from: https://www.plm.automation.siemens.com/global/en/our-story/glossary/digital-twin/24465.
[34] Beckhoff. TwinCAT Wind and oversampling technology enable highly efficient condition monitoring. [updated 2019 March]. Available from: https://www.beckhoff.com/media/downloads/applikationsberichte-downloads/2019/pcc_0319_goldwind_e.pdf.
[35] Ines Khelifi. Azure Digital Twins: Powering the next generation of IoT connected solutions. Updated [updated 2020 Jule 29]. Available from: https://www.plm.automation.siemens.com/global/en/our-story/glossary/digital-twin/24465.
[36] Ghezavati VR, Saidi-Mehrabad M. An efficient hybrid self-learning method for stochastic cellular manufacturing problem: a queuing-based analysis. Expert Syst Appl 2011;38(3):1326–35.
[37] Sharda B, Banerjee A. Robust manufacturing system design using multi objective genetic algorithms, Petri nets and Bayesian uncertainty representation. J Manuf Syst 2013;32(2):315–24.
[38] Ioannou G, Dimitriou S. Lead time estimation in MRP/ERP for make-to-order manufacturing systems. Int J Prod Econ 2012;139(2):551–63.
[39] Berling P, Farvid M. Lead-time investigation and estimation in divergent supply chains. Int J Prod Econ 2014;157(1):177–89.
[40] Qian X. System engineering. Shanghai: Shanghai Jiao Tong University Press; 2007.
[41] Estefan JA. Survey of model-based systems engineering (MBSE) methodologies. Incose MBSE Focus Group 2007;25(8):1–12.
[42] McGee TG, McGee BM. Agile systems engineering// systems engineering for microscale and nanoscale technologies. CRC Press; 2016. p. 86–119.
[43] Delligatti L. SysML distilled: a brief guide to the systems modeling language. Addison-Wesley; 2013.
[44] Aleksandraviciene A, Morkevicius A. MagicGrid book of knowledge. Vitae Litera, UAB; 2018.