Human-Computer Interaction & Emerging Technologies, Vol. 195, 2025, 399–405
https://doi.org/10.54941/ahfe1006257

Object-Oriented Encapsulation-Based Virtual Equipment Modeling for Digital-Twin Production Systems

Fengyi Feng and Xiaojun Liu

School of Mechanical Engineering, Southeast University, Nanjing, China

ABSTRACT

As digital-twin technology becomes integral to smart manufacturing, the creation of high-fidelity yet maintainable virtual equipment models is critical for effective production-unit debugging and system-level coordination. This paper presents an object-oriented encapsulation approach that leverages information-hiding and interface-uniformity mechanisms to modularize device states, interfaces, and behavior logic. By encapsulating each physical device within a self-contained class exposing only standardized input/output methods, the proposed method reduces coupling and enhances model reusability. We further introduce a hierarchical composition strategy, enabling seamless aggregation from single devices to work-unit, production-line, and workshop-level models. On top of this, we develop a semantic-signal aggregation framework and event-triggering mechanism that automatically translate low-level physical signals into discrete events for control and scheduling. A case study of a riveting workstation demonstrates improvements in interface consistency, modeling accuracy, and extensibility. The results confirm that our encapsulation-based modeling method offers a portable, scalable, and easily maintainable solution for digital-twin production systems, laying a solid foundation for advanced debugging workflows and intelligent decision support.

Keywords: Digital twin, Virtual equipment, Object-oriented encapsulation, Production unit composition, Semantic signal aggregation, Discrete event generation

INTRODUCTION

With the advancement of Industry 4.0, digital twin (DT) technology has become a key enabler for smart manufacturing, offering a real-time synchronized virtual representation of physical assets to support predictive analytics, operational optimization, and system resilience (Grieves and Vickers, 2017; Tao et al., 2019). However, in practical engineering applications, building scalable and maintainable virtual models remains challenging. Most virtual equipment models expose underlying implementation details and device-specific logic, resulting in high inter-module coupling and poor reusability (Lee et al., 2015; Fuller et al., 2020). Moreover, the raw signals generated by physical sensors are often low-level and continuous, lacking semantic clarity for higher-level reasoning and event-driven control.

Recent studies have further clarified the conceptual framework and reference architectures for DT-driven intelligent manufacturing (Lu et al., 2020), and have demonstrated practical frameworks for DT applications in high-precision sectors such as aeroengine blade production (Zhang and Zhu, 2019). Moreover, DT-driven cyber-physical systems (MCPS) have been proposed to support parallel control and decentralized decision-making in smart workshops under mass personalization paradigms (Leng et al., 2019).

To address these issues, this paper proposes an object-oriented modeling method based on encapsulation. The approach introduces two key mechanisms: information hiding and interface uniformity, which modularize internal states, sensor/actuator interfaces, and behavior logic of virtual devices. This reduces system complexity and enables plug-and-play reuse of equipment models. Additionally, we construct a semantic signal aggregation model and a rule-based event abstraction mechanism, allowing low-level physical data to be converted into high-level discrete events that drive task scheduling and logical inference.

Building on this, we design a multi-layer encapsulation framework, supporting recursive composition from virtual devices to production units, production lines, and entire workshops. A case study of a riveting workstation is conducted to validate the proposed method's effectiveness in terms of modeling fidelity, interface consistency, and system scalability. The results provide a foundation for deploying digital twins in complex manufacturing environments and for supporting intelligent debugging and decision-making processes.

OBJECT-ORIENTED ENCAPSULATION MODELING OF VIRTUAL EQUIPMENT

In digital twin systems, virtual equipment must accurately reflect the dynamic behavior and operational status of physical devices, respond to external inputs in real time, and provide consistent feedback. Due to the complexity and heterogeneity of physical equipment states and communication protocols, a direct, unstructured representation would hinder maintainability and scalability. Hence, adopting object-oriented encapsulation is essential.

Virtual equipment models encapsulate key attributes, internal states, and behaviors into self-contained classes that expose standardized input-output interfaces. As shown in Table 1, a typical encapsulated virtual equipment model consists of four fundamental components: internal state variables, sensor data interfaces, actuator control interfaces, and behavioral logic modules.

Table 1: Core components of an encapsulated virtual equipment model.

Component	Description	Example
Internal State	Real-time operational parameters	Speed, position, temperature
Sensor Interface	Input ports or methods for receiving sensor data	Pressure sensor, displacement sensor

Continued

Table 1: Continued

Component	Description	Example
Actuator Interface	Methods for controlling actuators	Start, stop, move, rivet
Behavioral Logic	Internal rules or methods encoding control logic	Execute next step upon condition, signal on threshold

CORE ENCAPSULATION MECHANISMS

The encapsulation framework relies on two key mechanisms: information hiding and interface uniformity, which jointly support the modularity, independence, and interoperability of virtual device models.

Information hiding isolates implementation details—such as state variables, algorithms, and control logic—from external access. Internal attributes like sensor values or temporary states are exposed only through public methods (e.g., getStatus()), ensuring external modules are decoupled from internal data structures. Behavioral logic (e.g., start(), stop()) is encapsulated within the class, allowing changes without affecting dependent modules.

Interface uniformity mandates standardized functional interfaces for all device classes. For example, all sensors implement readSensor(), while all actuators support executeCommand() or start()/stop() methods. This consistency enables plug-and-play integration, simplifies system calls, and supports model reuse and automation.

Together, these two mechanisms provide both encapsulation boundaries and standardized communication channels, ensuring scalability, reusability, and maintainability in complex digital twin systems.

ENCAPSULATION WORKFLOW FOR VIRTUAL EQUIPMENT

The development of an encapsulated virtual equipment model generally follows a top-down modeling workflow, which ensures structural clarity, behavioral accuracy, and interface standardization throughout the process:

Step 1: Analyze the structure and function of the physical device.

This step involves systematic identification of the device's subcomponents (e.g., sensors, actuators, control elements) and clarification of their roles and dynamic characteristics. It establishes the foundation for accurate virtual representation.

Step 2: Define internal states and key attributes.

Based on functional analysis, core state variables—such as position, pressure, temperature, or operation flags—are defined to reflect the physical equipment's status under various working conditions. These variables will be encapsulated as internal private fields.

Step 3: Design standardized interfaces.

Sensor inputs and actuator commands are abstracted into standardized methods (e.g.), providing a clear and consistent communication interface. This step ensures that each device can interact seamlessly with others through uniform input/output channels. readSensor()executeCommand().

Step 4: Encapsulate behavioral logic.

The equipment's control logic—such as motion sequences, condition checks, or output triggers—is implemented as internal methods of the virtual class. These methods remain hidden from external modules, enforcing modular separation and improving maintainability.

Step 5: Validate and iteratively refine the model.

The virtual model is tested against simulation results or real-device data to verify its correctness in terms of state evolution and response accuracy. Model parameters and behavioral rules are adjusted iteratively to ensure fidelity and robustness under different scenarios.

PRODUCTION UNIT COMPOSITION AND HIERARCHICAL ENCAPSULATION

To enable structured and scalable digital twin modeling, the encapsulated virtual equipment models must be further abstracted and composed into higher-level production entities. This section introduces the composition of production units and hierarchical encapsulation strategies, which extend the object-oriented encapsulation principle from individual devices to complete production lines and workshops.

At the production unit level, multiple encapsulated virtual devices are integrated to accomplish a defined functional task—such as riveting, assembly, or inspection—through modular composition. Each unit exposes abstract functional interfaces to the external system, such as startWorkUnit(), stopWorkUnit(), and getUnitStatus(). These interfaces focus on the overall functional boundaries of the production unit, rather than exposing individual device operations. For instance, in a riveting workstation scenario, the entire unit encapsulates devices like the fixture controller, riveting machine, and conveyor, with the goal of executing a complete "riveting task" without revealing the internal coordination details.

Internally, these virtual devices interact through well-defined control sequences and data exchange mechanisms. For example, the fixture controller first performs clamping and alignment; the riveting machine then executes the pressing action; and finally, the conveyor transfers the workpiece to the next station. These interactions are coordinated by an internal management class—WorkUnitController—which encapsulates execution logic, device invocation order, and exception handling routines. This controller governs the process flow and synchronizes the submodules based on event notifications and interface calls, ensuring that the entire unit behaves as a cohesive, reusable module. External systems can invoke the unit without needing to manage the detailed internal device dependencies.

Building on production unit composition, hierarchical encapsulation further abstracts and aggregates multiple units into higher-level organizational structures. For example, several production units can be encapsulated into a ProductionLine model, which in turn may be integrated into a Workshop model representing a full manufacturing section. Each level exposes only its functional control interfaces, such as startLine() or getLineStatus(), while hiding the internal composition of subordinate units. This recursive abstraction structure ensures that system designers and controllers can operate at varying levels of granularity, from managing single devices to orchestrating entire production lines, without being burdened by low-level detail.

This layered encapsulation strategy forms the foundation for modular, maintainable, and large-scale digital twin modeling in real-world manufacturing environments.

SEMANTIC SIGNAL AGGREGATION AND EVENT ABSTRACTION

Raw sensor data in digital twin environments are often noisy and semantically weak. We propose a method to aggregate low-level signals into meaningful semantic indicators and convert them into discrete events for system-level processing.

Within each virtual device, raw signals are filtered, classified, and fused according to predefined rules and operational context. For instance, when the displacement reaches a threshold and the pressure exceeds a limit, the system recognizes the semantic state "Riveting Completed." This high-level semantic signal is then monitored by an internal rule engine. When conditions are met, the device emits standardized events (e.g., RIVET_DONE) with timestamps and metadata. These events drive upper-level control logic without exposing signal-level complexities.

This mechanism enables the use of event-driven architectures for digital twin-based virtual commissioning, improving system responsiveness, logic clarity, and scalability.

CASE STUDY: VIRTUAL EQUIPMENT MODELING OF A RIVETING WORKSTATION

To validate the proposed encapsulation-based modeling method, we conducted a full-scale implementation of a virtual model for a typical industrial riveting workstation. The physical workstation consists of three main components: a hydraulic riveting machine, a fixture controller, and a conveyor mechanism. These devices collaboratively execute a riveting task that involves workpiece clamping, pressing, and part transfer.

As shown in Figure 1, the riveting machine is modeled as a virtual equipment class named RivetMachine, which encapsulates core components such as the pressure sensor, displacement sensor, hydraulic cylinder, riveting head, and clamping fixture. Key internal states—including real-time pressure and displacement—are stored as private attributes. Sensor interfaces such as readPressure() and readDisplacement(), and actuator methods like startRivet() and stopRivet() are defined as standardized public interfaces. Within the method startRivet(), behavioral logic is embedded to continuously monitor sensor data and evaluate whether both the pressure and displacement have reached predefined thresholds. Once these conditions are satisfied, the device automatically updates its internal state and emits a semantic event RIVET_DONE to the upper-level controller.

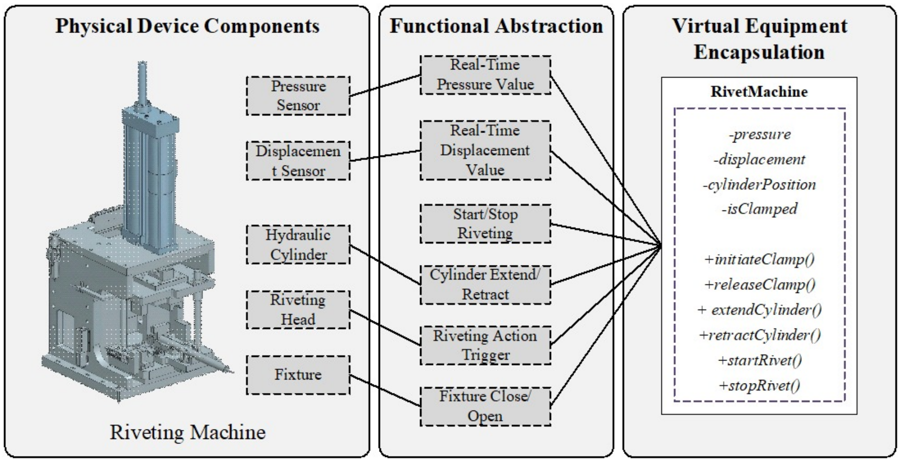

Figure 1: Encapsulation process of the riveting machine virtual equipment.

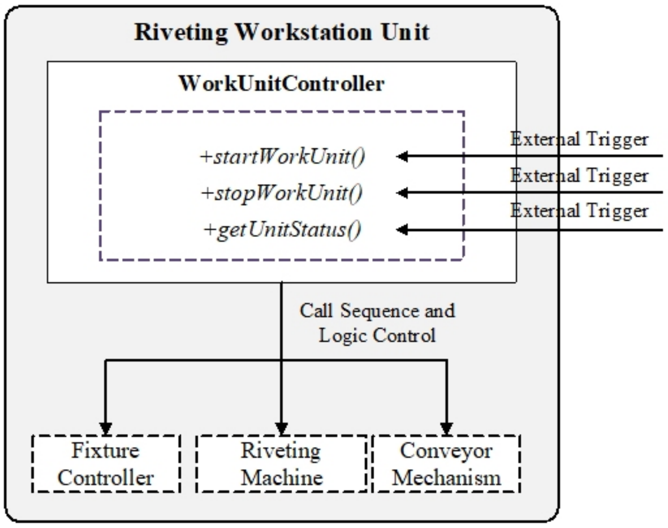

Figure 2: Schematic diagram of the riveting unit composition.

To extend beyond the single device, we encapsulate the entire workstation as a production unit model named RivetWorkUnit. As illustrated in Figure 2, this unit consists of three virtual device submodules (the fixture controller, riveting machine, and conveyor), coordinated by a central management class called WorkUnitController. The controller governs execution logic through event-driven coordination. Upon invocation of the startWorkUnit() method, the system initiates the clamping process via the fixture controller. Once the clamping is confirmed via the event CLAMP_DONE, the riveting machine is triggered to execute the riveting task. After the RIVET_DONE event is received, the conveyor is activated to transfer the completed part to the next station. Throughout the process, the controller also handles abnormal conditions through fault signals (FAULT) and can terminate operations via stopWorkUnit() in response to system-level exceptions.

This case demonstrates that the proposed encapsulated modeling method effectively supports not only high-fidelity device simulation, but also modular unit composition and logic coordination. The model was calibrated iteratively against real equipment data to ensure accuracy, and experiments showed that the encapsulated structure allows new equipment to be substituted or reconfigured without altering upper-level coordination logic—demonstrating its high reusability and maintainability in real-world applications.

CONCLUSION

This paper proposes an object-oriented encapsulation approach to virtual equipment modeling in digital-twin production systems, addressing the issues of high coupling, poor scalability, and difficulties in bridging low-level signals to high-level logic. Leveraging the mechanisms of information hiding and interface uniformity, our method modularly encapsulates device states, interfaces, and internal logic, thereby effectively reducing inter-module dependencies and enhancing flexibility and maintainability. Additionally, we introduce a semantic signal aggregation and event abstraction framework, transforming raw sensor data into high-level semantic signals and discrete events, significantly improving the logical reasoning and scheduling capability within digital-twin systems. A comprehensive case study of a riveting workstation confirmed the practicality of the proposed method in real-world manufacturing contexts, demonstrating high fidelity in equipment simulation, rapid device integration, and excellent scalability. Future research directions include validating and refining the proposed method in more complex multi-unit and cross-platform digital-twin scenarios, as well as exploring automated encapsulation modeling and event-rule generation techniques to enhance generality and intelligence.

REFERENCES

Fuller, A., Fan, Z., Day, C. & Barlow, C. 2020. Digital twin: Enabling technologies, challenges and open research. *IEEE access,* 8, 108952–108971.

Grieves, M. & Vickers, J. 2017. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. *Transdisciplinary perspectives on complex systems: New findings and approaches*, 85–113.

Lee, J., Bagheri, B. & Kao, H.-A. 2015. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters,* 3, 18–23.

Leng, J. W., Zhang, H., Yan, D. X., Liu, Q., Chen, X. & Zhang, D. 2019. Digital twin-driven manufacturing cyber-physical system for parallel controlling of smart workshop. *Journal of Ambient Intelligence and Humanized Computing,* 10, 1155–1166.

Lu, Y. Q., Liu, C., Wang, K. I. K., Huang, H. Y. & Xu, X. 2020. Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues. *Robotics and Computer-Integrated Manufacturing,* 61.

Tao, F., Zhang, M. & Nee, A. Y. C. 2019. *Digital twin driven smart manufacturing*, Academic press.

Zhang, X. Q. & Zhu, W. H. 2019. Application framework of digital twin-driven product smart manufacturing system: A case study of aeroengine blade manufacturing. *International Journal of Advanced Robotic Systems,* 16.