Enabling Non-Simulation Experts to Adapt Digital Twins during Production System Redesign using Agentic AI

Christian Schwede
University of Applied Sciences and Arts Bielefeld
Bielefeld, Germany
Industrial Manufacturing
Fraunhofer Institute for Software and Systems Engineering ISST
Dortmund, Germany
christian.schwede@hsbi.de

Lennart Schipper
University of Applied Sciences and Arts Bielefeld
Bielefeld, Germany
lennart.schipper@hsbi.de

Jan Cirullies
Dortmund University of Applied Sciences and Arts
Dortmund, Germany
Industrial Manufacturing
Fraunhofer Institute for Software and Systems Engineering ISST
Dortmund, Germany
jan.cirullies@fh-dortmund.de

Abstract

Simulation-based digital twins are a core foundation for the efficient and holistic optimization of material-flow systems across the lifecycle – from design and planning to control. However, constructing digital representations of supply chains and production/logistics systems is labor-intensive and largely manual. Current research seeks to shift from manual expert modeling to auto-generated digital twins derived from event-log data using process mining and machine learning. Yet, structural adaptation of these models during (re)design typically still requires intervention by simulation experts. This paper presents an approach that enables structural adaptation of digital-twin models during redesign via an LLM-based design assistant, which performs modeling through natural-language interaction with non-expert users. The assistant employs an agentic, retrieval-augmented generation (RAG) workflow and relies exclusively on publicly available information from the open-source digital-twin platform OpenFactoryTwin (OFacT) to execute adaptations. The proposed framework is evaluated in an empirical study based on data from an IoT-Factory - a laboratory assembly environment.

CCS Concepts

• **Applied computing** → Enterprise computing; Enterprise modeling; • **Information systems** → Information systems applications; Decision support systems; Expert systems; • **Human-centered computing** → Human computer interaction (HCI); Interaction paradigms; Natural language interfaces.

Keywords

Digital twins, Production system design, Material flow, Large language models

ACM Reference Format:
Christian Schwede, Lennart Schipper, and Jan Cirullies. 2026. Enabling Non-Simulation Experts to Adapt Digital Twins during Production System Redesign using Agentic AI. In *2026 9th International Conference on Computers in Management and Business (ICCMB 2026), February 27–March 01, 2026, Tokyo, Japan.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3802463.3802479

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
ICCMB 2026, Tokyo, Japan
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2228-8/2026/02
https://doi.org/10.1145/3802463.3802479

1 INTRODUCTION

Simulation-based digital twins (SBDT) have been vastly applied to optimize material flow systems (MFS) including production systems, warehouses and supply chains across the design, planning and control phases [1]. For supply chains, digital twins promise substantial benefits, e.g., "improving customer experience" and "reducing costs", yet Chief Supply Chain Officers see "lack of required digitalization skills" and "investment costs" as critical barriers [2]. Consequently, recent research has shifted toward data-driven, automatically generated SBDTs learned from operational MFS data [3]. In particular, standardized event-log data commonly employed in process mining have been pivotal for automatic generation due to their consistent format and broad availability [4]. While auto-generating SBDT works well for existing MFS, structural design and redesign remain the domain of simulation experts. Prior methods – such as standardized simulation building blocks, metamodels, and guided interviews – can reduce manual effort, but integrating them into an existing digital-twin framework still entails substantial overhead.

In our recent work, we introduced an LLM interface between the production planner and the SBDT [5]. Although the assistant performed adaptations robustly, it still required the creation of so-called framework-specific step instructions for every single adaptation type to mediate between the LLM and the digital-twin platform. Authoring these instructions can only be done by simulation experts and, even if required only once per framework, introduces additional manual effort.

To remove this bottleneck, this paper presents an approach that uses an agentic AI to perform adaptations by leveraging only publicly available information – including documentation and source code – of the open-source OFacT framework. This eliminates the need for expert-authored, framework-specific step instructions while preserving correctness and traceability.

2 RELATED WORK

2.1 Reducing the Effort for Modelling Digital Twins of Production Systems

The term "digital twin," formalized by NASA in 2010, builds on earlier concepts. Since then, multiple definitions have emerged that emphasize different dimensions of the technology [6]. Kritzinger et al. [1] differentiate: a digital model with manual data flow, a digital shadow with unidirectional flow, and a digital twin with bidirectional exchange. Here, a shadow of an existing production line is adapted to design a new line, evolving into a full digital twin during commissioning and operation. Despite their central role in digital transformation, modelling and deployment costs remain significant barriers [1, 6].

Reducing the effort required to develop the digital-twin or simulation model (for SBDTs) is a primary lever to address these challenges. Four categories could be identified:

- Data-driven model generation: methods that automate modelling by inferring the model (or its components) directly from operational data, without relying on reference models. While effective for many SBDT components, these methods are constrained by data availability and quality. They struggle synthesizing complete system models – particularly for complex configurations such as matrix production systems [3].
- Model reuse via building blocks: approaches that compose models from pre-existing, manually curated components (building blocks) maintained in libraries or repositories. Their effectiveness depends on the availability, granularity, and interoperability of suitable blocks [7, 8].
- Hybrid approaches combining building-block-based modelling with automatic model generation, often via multiple variants. Automatic generation can be categorized into: (i) parametric approaches that assemble models from building blocks and configure them via parameters [9]; (ii) structural approaches that derive models from system structure using data from external sources (e.g., CAD systems and factory layout models) [10]; and (iii) hybrid, knowledge-based approaches that combine artificial intelligence with parametric and structural methods [11].
- Parameterized models coupled with search/optimization algorithms to identify suitable configurations [12].

All approaches reduce effort but still require expert knowledge to select, adapt, or develop models and blocks. Moreover, even data-driven approaches depend on existing data, limiting applicability to new production lines and leaving the specification of requirements and constraints to experts.

2.2 Expert Knowledge Utilization

As noted in 2.1, no approach fully eliminates the need for simulation expert knowledge in design-phase modelling. Production expertise must be made accessible to modelling pipelines or experts enabled to build models themselves. Jungmann and Lazarova-Molnar [13] present a method to incorporate expert knowledge into data-driven digital twins, addressing knowledge not captured in operational data. The approach requires explicit formalization of expert knowledge, which is then integrated via rule-based algorithms into Petri-net models used for model generation.

While effective, the need for rigorous formalization remains a significant barrier. Advances in large language models (LLMs) and transformer architectures enable effective processing of natural language, supporting context-aware modeling and reducing dependence on formalized representations [14]. Studies show that LLMs can synthesize conceptual and process models – such as ER, BPMN, and UML – from textual specifications, with iterative feedback improving reliability [15, 16]. Schipper and Schwede further demonstrate that coupling an LLM with a digital twin allows production experts to adapt existing models with less effort, though reliance on predefined step instructions remains a limitation [5].

2.3 AI Agents and Agentic Systems

LLM-based, iterative systems (cf. 2.2) are commonly referred to as AI agents or agentic AI systems. The AI agents leverage LLMs to autonomously pursue goal-directed tasks and are deployed in domains such as automation, customer service, and decision support through tool use and interaction with external environments [17]. Although they exhibit autonomy, task specificity, and reactivity, they face challenges including hallucination and stale knowledge. Multi-agent (the agentic AI) architectures mitigate these issues through collaboration between multiple agents for different subtasks [17]. Recent work improves planning efficiency via graph-based task execution [18] and adaptive task decomposition (e.g., ADaPT), which increases success rates without additional LLM calls [19] yet tool integration remains critical; systems employing Python-based actions for flexible execution and self-debugging outperform traditional JSON/XML interfaces [20].

Building on our prior work linking agentic systems to digital twins for model adaptation, this paper moves from AI agents to agentic AI and replaces predefined step instructions by employing retrieval-augmented generation (RAG), thereby reducing modelling effort and increasing design-time flexibility for adapting digital-twin models.

We build on Open Factory Twin (OFacT), an open-source, Python-based digital-twin framework for production and logistics that supports operational monitoring and simulation-based design of MFS [21]. OFacT is designed for data-driven, auto-generation of digital twins from process data. The framework comprises (i) a static configuration space (state model) defining entities, resources, and processes, and (ii) a dynamic behavior space capturing real or simulated process executions. Historically, modelling relied on an Excel interface. This paper introduces a new modelling workflow that enables production experts to adapt digital twins more intuitively. The case studies apply OFacT's automated event-log-based model generation pipeline.

3 LLM-BASED DESIGN ASSISTANT

The proposed framework adapts digital twins using an agentic AI and enhances the framework by Schipper and Schwede [5], including four components: the digital twin software (DTS), the digital twin software interface (DTSI), the agentic system, and the RAG pipeline. While the DTS remains largely the same, the DTSI

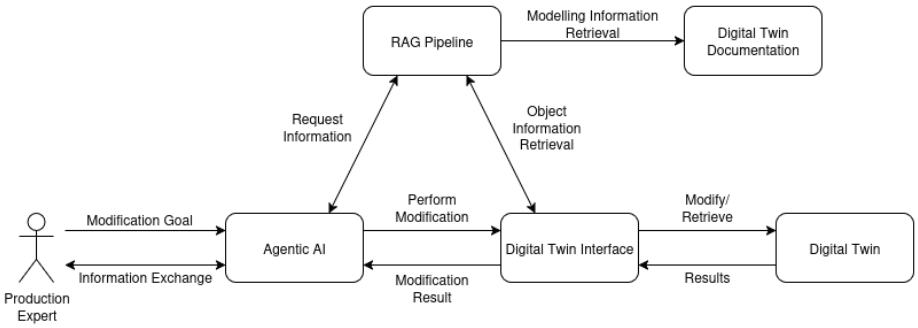

Figure 1: Architecture of the framework for adapting SBDTs using an agentic AI and RAG.

was redesigned for improved flexibility, error handling, and easier maintenance. The key advancements include replacing the previous AI agent with an agentic AI architecture and substituting step instructions with a RAG pipeline, making the framework applicable to digital twins of any structure rather than only those generated via process mining (see Figure 1).

Implemented with the Smolagents framework, the agentic system comprises agents and their environment and supports both flexible code agents and more restricted but reliable tool agents. A hierarchical design employs five specialized code agents: The main agent coordinates and delegates tasks, while the other four agents handle the operations of deleting, merging, modifying of model elements as well as summarizing the overall task. Objects are created separately through a dedicated Create Object tool agent for reliability reasons. This design yields role specialization, reduced tool complexity, and simpler task planning, at the cost of higher system and communication complexity. Communication and reasoning follow standardized templates and the ReAct pattern (reasoning, action, observation) to improve adaptability and reduce hallucinations.

Retrieval-augmented generation improves accuracy by supplying retrieved context to the LLM, mitigating hallucinations and stale knowledge [22]. The framework uses an adaptive, modular pipeline that dynamically decides when and how to retrieve, with modules including a router, retriever (vector- or graph-based), query rewriter, generator, post-processor, judge, and a specialized Find Object module for interacting with the digital twin's state model. Documentation and state-model data are preprocessed via structured indexing and embeddings – leveraging MkDocs-specific structures (e.g., example and note blocks) – to enable efficient retrieval and ensure flexibility and context awareness across diverse digital twin configurations.

4 EVALUATION OF THE AGENTIC-AI-BASED ARCHITECTURE

4.1 Evaluation Scenarios and Design Variants

We evaluate using a model of the IoT-Factory at the University of Applied Sciences and Arts Bielefeld (HSBI). The factory assembles a smart product (printed circuit board, small display, case) in ~150 variants across three fully automated assembly stations, with inline quality tests after each station as well as mobile robots for parts supply. It supports automated assembly and disassembly without human intervention. The SBDT was auto-generated from factory event logs using OFacT.

Two complementary assessments were conducted. First, two adaptation scenarios from the IoT-Factory case were used to compare alternative framework design choices. Scenario 1 required the agentic system to add a feature (e.g., humidity sensor) to an existing feature cluster (e.g., sensors) – thus update the product structure – and adjust selection probabilities to 100%, challenging LLM abilities to handle calculations. Scenario 2 required adding a new assembly station with associated parts and process execution plans, consolidating properties from existing stations, and optionally removing legacy stations which challenges complexity and correct references between various changes. Each scenario was executed five times per design choice. Metrics included success rate, total token usage, and peak context size. Token usage and context size serve as indirect indicators of efficiency and runtime.

Each of the two scenarios (new feature and new assembly station) was executed in three different design variants:

Variant 1: agent cardinality. Application of one, few or many agents coordinating the adaptations

Variant 2: planning methodology. ADaPT planner (dedicated planning agent with dynamic task decomposition) compared to a RAG-driven planner (plans synthesized directly from retrieved context).

Variant 3: model class. foundation LLMs vs. reasoning-enhanced LLMs

This resulted in 14 evaluated scenarios.

Second, we reused five scenarios of increasing complexity from our prior work to compare the previous and the new framework. These scenarios include adding a new feature, adding a stationary resource, modifying a transition model, creating a value-added process, and a composite operation combining all prior modifications. We measured total modification time (end-to-end adaptation duration) and user interaction time (time spent composing prompts and reviewing responses) and success rates. All tests used the Qwen3-32B model.

Table 1: Results of the design decision scenarios.

Design Variant	Scenario	Design Variant Characteristic	Success Rate	Max. Context Size	Token Usage
Agent cardinality	Feature	One Agent	0.4	11,504	140,324
		Few Agents	**0.8**	11,705	217,643
		Many Agents	0.4	12,108	502,752
	Assembly station	One Agent	0.0	-	-
		Few Agents	**0.8**	16,089	427,790
		Many Agents	0.6	19,662	994,392
Planning methodology	Feature	ADaPT	**0.8**	13,667	422,284
		RAG Plan	**0.8**	12,093	414,943
	Assembly station	ADaPT	0.4	19,845	767,868
		RAG Plan	**1.0**	14,853	390,852
Model class	Feature	Reasoning	0.6	11,614	235,826
		No-Reasoning	**1.0**	10,312	206,174
	Assembly station	Reasoning	0.2	18,279	226,188
		No-Reasoning	**0.8**	16,657	496,515

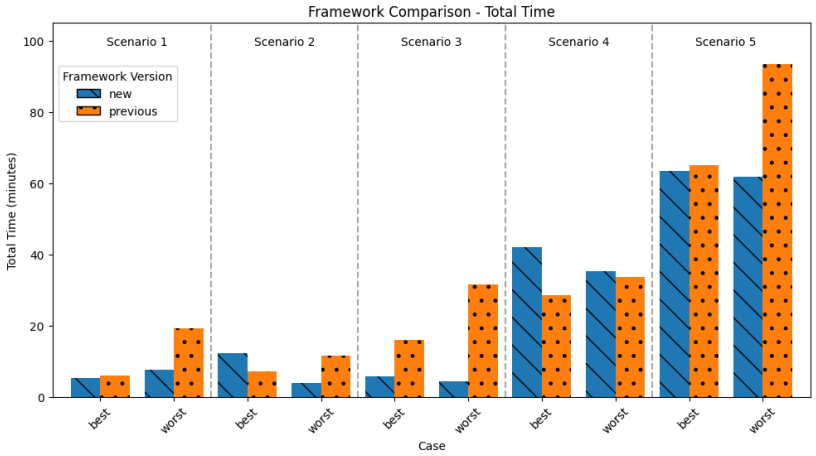

Figure 2: Total modelling time comparison between AI agent and RAG -based framework.

4.2 Results from the Evaluated Scenarios for Design Choices

Table 1 summarizes the results of all 14 scenario executions. Variant 1 (agent cardinality) showed that among multi-agent designs, a lean setup with specialized tools (cf. section 3) achieved the highest success rates, whereas replacing those tools with additional agents reduced success due to coordination overhead. Variant 2 (planning methodology) showed that both a modified ADaPT planner and an RAG-driven planner performed similarly on the simple scenario, but the RAG-based approach achieved higher success rates and more efficient context usage on the complex scenario; it was therefore selected for the final framework. Variant 3 (model class) showed that foundation LLMs outperform reasoning-enhanced LLMs in both scenarios, delivering higher success through greater reliability and fewer unnecessary actions. Deliberative models more often deviated from plans, consumed more tokens, and underperformed in the agentic setting.

4.3 Results from Comparison with Former Framework

Comparisons of user-interaction and total modification times across the five scenarios show negligible differences between the two frameworks. User-interaction times were similar except in Scenario 5 – the most complex case, combining all prior modifications in a single session – where the new framework required more user time, likely due to an additional validation step to ensure alignment with the user's intent (Figure 2).

To evaluate success rate, the two scenarios ((1) adding feature and (2) adding value added process) were executed five times each to cover a range of different complexities. For the simpler (1) scenario both achieved 100% success while for the more complex (2) scenario the former framework achieved slightly higher reliability (80% vs. 60%), plausibly because detailed step instructions constrained the agent's behavior, reducing flexibility but also modelling errors.

5 SUMMARY AND OUTLOOK

To enable supply chain and production planners to directly adapt SBDTs during (re)design, we propose shifting from a single-agent LLM to an agentic AI, thereby increasing flexibility and minimizing DT-specific engineering (e.g., step instructions). The evaluated framework achieves high adaptation success rates with substantially fewer DT-specific requirements.

Nonetheless, limitations remain: (i) the current design still relies on specialized tools (e.g., object creation, reference retrieval) that could be integrated into agent behaviors; and (ii) further evaluation on real-world cases with production planners is needed to mitigate developer bias in prompt formulation.

Acknowledgments

The research in the project "DaCoR – Data-driven Learning of Control Rules" was funded by the Center of Excellence Logistics and IT (Leistungszentrum Logistik und IT).

References

[1] Kritzinger, W., *et al.* 2018. Digital twin in manufacturing: A categorical literature review and classification. IFAC-PapersOnLine 51, 11, 1016–1022.
[2] Gartner. 2025. Supply chains building a digital twin of the customer. Abgerufen am 8. Oktober 2025 von https://emt.gartnerweb.com/ngw/globalassets/en/supply-chain/documents/digital-twin-of-the-customer.pdf.
[3] Schwede, C., and Fischer, D. 2024. Learning simulation-based digital twins for discrete material flow systems: A review. In Proceedings of the 2024 Winter Simulation Conference (WSC). 3070–3081.
[4] Friederich, J., Francis, D. P., Lazarova-Molnar, S., and Mohamed, N. 2022. A framework for data-driven digital twins of smart manufacturing systems. Computers in Industry 136, 103586.
[5] Schipper, L., and Schwede, C. 2025. Automated creation of digital twins for production line design by adapting existing models through LLM-guided expert interviews. In Simulation in Produktion und Logistik 2025. 149–158.
[6] Singh, M., *et al.* 2021. Digital twin: Origin to future. Applied System Innovation 4, 2, 36.
[7] Horbach, S., *et al.* 2011. Building blocks for adaptable factory systems. Robotics and Computer-Integrated Manufacturing 27, 4, 735–740.
[8] Gössler, G., und Sifakis, J. 2005. Composition for component-based modeling. Science of Computer Programming 55, 1, 161–183.
[9] Kallat, F., *et al.* 2020. Using component-based software synthesis and constraint solving to generate sets of manufacturing simulation models. Procedia CIRP 93, 556–561.
[10] Huang, Y., Seck, M. D., und Verbraeck, A. 2011. From data to simulation models: Component-based model generation with a data-driven approach. In Proceedings of the 2011 Winter Simulation Conference (WSC). 3719–3729.
[11] Bergmann, S., und Strassburger, S. 2010. Challenges for the automatic generation of simulation models for production systems. In Proceedings of the 2010 Summer Computer Simulation Conference (SCSC '10). Society for Computer Simulation International, San Diego, CA, USA, 545–549.
[12] Wang, G., *et al.* 2008. A simulation optimization approach for facility layout problem. In 2008 IEEE International Conference on Industrial Engineering and Engineering Management. 734–738.
[13] Jungmann, M., und Lazarova-Molnar, S. 2024. Fusing expert knowledge and data for simulation model discovery in digital twins: A case study from reliability modeling. In Proceedings of the 2024 Winter Simulation Conference (WSC). 2463–2474.
[14] Karanikolas, N., *et al.* 2024. Large language models versus natural language understanding and generation. In Proceedings of the 27th Pan-Hellenic Conference on Progress in Computing and Informatics (PCI '23). Association for Computing Machinery, New York, NY, USA, 278–290.
[15] Fill, H.-G., Fettke, P., und Köpke, J. 2023. Conceptual modeling and large language models: Impressions from first experiments with ChatGPT. Enterprise Modelling and Information Systems Architectures (EMISAJ) – International Journal of Conceptual Modeling 18, 3, 1–15.
[16] Kourani, H., *et al.* 2024. Process modeling with large language models. arXiv: 2403.07541.
[17] Sapkota, R., Roumeliotis, K. I., und Karkee, M. 2025. AI agents vs. agentic AI: A conceptual taxonomy, applications and challenges. arXiv:2505.10468.
[18] Gabriel, A. G., Alameer Ahmad, A., und Jeyakumar, S. K. 2024. Advancing agentic systems: Dynamic task decomposition, tool integration and evaluation using novel metrics and dataset. arXiv:2410.22457.
[19] Prasad, A., *et al.* 2024. ADaPT: As-needed decomposition and planning with language models. arXiv:2311.05772.
[20] Wang, X., *et al.* 2024. Executable code actions elicit better LLM agents. arXiv: 2402.01030.
[21] Schwede, C., und Freiter, A. 2024. OpenFactoryTwin (OFacT): Initial release (0.1.0). Zenodo.
[22] Gao, Y., *et al.* 2024. Retrieval-augmented generation for large language models: A survey. arXiv:2312.10997.