# Digital Twin Builder: A Multi-Agent LLM System for Automated Industrial Digital Twin Development

**Maria Dziuba** [1] **Elizaveta Egorova** [1] **Nikolay Vasilev** [1] **Ilya Novitskiy** [1] **Vladislav Tuchin** [1] **Valeria Efimova** [1]

## Abstract

Multi-agent LLM systems are increasingly applied to complex code generation tasks, including industrial software, but most existing systems are narrow, lack safety guaranties, and cannot be reused across production scenarios. We present Digital Twin Builder, an end-to-end multi-agent system that builds industrial digital twins (DTs) from natural-language descriptions of manufacturing processes. An orchestrator coordinates specialized agents for user interaction, schema design, code generation, and a multi-modal complex anomaly detector that fuses sensor and camera streams. Safety is enforced via schema-grounded synthesis with structural and functional test gates at evaluation time, plus bounded fail-closed orchestration at runtime. On a steel-casting/cement-plant benchmark, our pipeline attains $100\%$ Structural Test Rate, $5.5\times$ over a single-LLM ablation, within 10 J-LLM points of Claude 4.6 Sonnet; a blind study over 118 scenarios confirms decomposition as the main driver of functional completeness. Our code is publicly available at `https://github.com/CTLab-ITMO/DigitalTwinBuilder`.

## 1. Introduction

Multi-agent systems built on top of large language models (LLMs) are now applied across a remarkable range of domains, from software engineering and scientific workflows to robotics and, increasingly, industrial automation (Reinpold et al., 2025; Nie et al., 2022). By decomposing a complex task into specialized roles coordinated by an orchestrator, such systems can plan, write and validate code that no single monolithic LLM reliably produces in one pass. As these architectures move from generic coding assistants towards real production environments, the requirements for the generated artifacts change qualitatively: in industrial settings, code is not just a draft to be polished by a human, but a component that ultimately drives sensors, controllers, and entire manufacturing lines.

In production, therefore, generated code must be safe (verifiable, schema-grounded, testable), general (reusable across heterogeneous assets and modalities), and end-to-end (translating informal user intent into a running system) — properties that narrow, task-specific multi-agent pipelines do not jointly provide. Most existing systems target a single sub-problem, rely on hand-crafted glue code, or assume a fixed asset model, and they offer only weak guarantees that the resulting program is structurally and functionally correct before it is deployed against physical equipment.

We pursue a general, safety-oriented multi-agent system for industrial code generation, and demonstrate it through one of its most demanding instantiations: the automatic construction of industrial digital twins (DTs) (Preuveneers et al., 2018; Mihai et al., 2021). Digital twins are an attractive testbed because they require simultaneously a formal description of the asset, executable simulation and control code, and a live coupling to sensor and camera streams; getting any of these pieces wrong silently breaks the link between the virtual and the physical system. A multi-agent pipeline that can build such twins from a natural-language description must therefore combine schema design, code synthesis, multi-modal perception, and runtime orchestration in a single, auditable workflow.

Existing LLM-based efforts in this space cover only isolated slices of the problem — AAS generation (Xia et al., 2024b), device-code synthesis (Zhao et al., 2026), and simulation parametrization (Xia et al., 2024a) — still requiring manual interface construction (Xia et al., 2023; Gebreab et al., 2024). Automated schema design with schema-grounded code generation is absent, and multi-modal sensor and IP-camera integration via GigE Vision (Automated Imaging Association, 2020), RTSP (Schulzrinne et al., 1998), and ONVIF (ON-

[1]ITMO University, Saint Petersburg, Russia. Correspondence to: Maria Dziuba <dziuba.maria@niuitmo.ru>, Elizaveta Egorova <elvegorova@itmo.ru>, Nikolay Vasilev <nvasilev@itmo.ru>, Ilya Novitskiy <367443@niuitmo.ru>, Vladislav Tuchin <vl_tuchin@itmo.ru>, Valeria Efimova <vefimova@itmo.ru>.

*Proceedings of the $43^{rd}$ International Conference on Machine Learning*, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

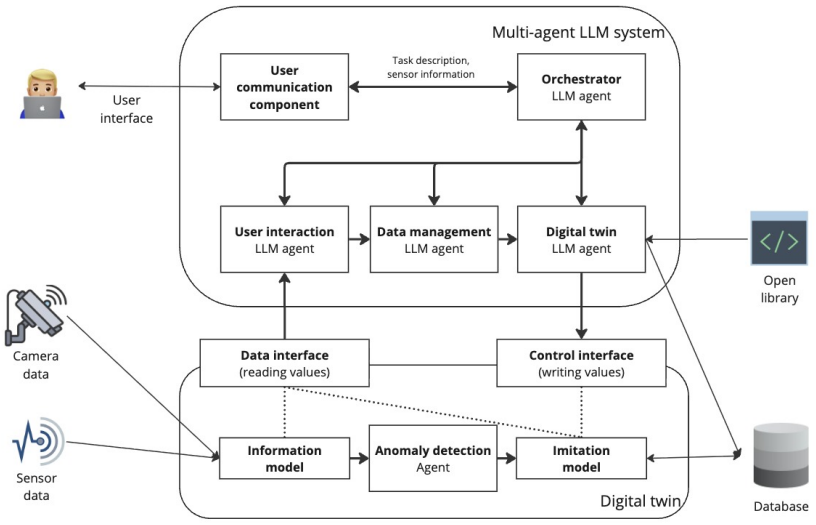


*Figure 1.* Schema of the multi-agent system for the automated development of a production digital twin.

VIF, 2024) protocols remains underdeveloped (Wang & Qin, 2024). Industrial DT development is dominated by proprietary platforms (Amazon Web Services, 2024; Microsoft, 2025; Siemens, 2025; PTC, 2025; Ansys, 2025; AVEVA, 2025), while open-source OpenTwins (Robles et al., 2023) on Eclipse Ditto (Eclipse Foundation, 2024) still defines DTs manually via REST. MAS4DT (Pretel et al., 2025) proposes a Prometheus-based methodology (Padgham & Winikoff, 2004) without implementation; the paper (Yang et al., 2025b) builds a ChatDev-style (Qian et al., 2023) MAS validated only in simulation. Task-specific LLM systems address isolated slices (Gebreab et al., 2024; Holt et al., 2024; Lee et al., 2025; Xia et al., 2024a; Alexopoulos et al., 2025; Amad et al., 2025; Barat et al., 2025; Lin et al., 2025; Mandal & O'Connor, 2024), but none simultaneously offer (a) end-to-end manufacturing DT automation, (b) open-source availability, (c) multi-agent LLM control, and (d) integrated sensor, as well as camera and code-generation support.

We make the following contributions:

1. A general, safety-oriented multi-agent architecture for industrial code generation, with an orchestrator routing work to task-specific LLMs, schema-grounded synthesis, and structural and functional test gates plus bounded fail-closed orchestration at runtime.
2. An end-to-end automation pipeline, instantiated as Digital Twin Builder, that turns a natural-language description of a manufacturing process into a runnable DT with relational schema, simulation code, IoT and IP-camera integration, and an online anomaly-detection loop without manual interface construction.

## 2. Methodology

Digital Twin Builder (Figure 1) instantiates the multi-agent architecture described above: an orchestrator routes work to four specialized agents, and an open-source library connects sensors and IP cameras over GigE Vision, RTSP, and ONVIF.

**Orchestrator agent.** A Nemotron-Orchestrator-8B (NVIDIA, 2025) mediates every interaction. The four agents and three low-level capabilities (`ipcamera_gige`, `ipcamera_rtsp`, `sensor_manager`) are exposed as JSON-schema tools alongside `finish`. The orchestrator plans inside a `<think>` block, emits strict-JSON tool calls, and consumes responses until `finish` or a step budget. It is tuned with an Online RL reward (sim/real divergence penalty, per-tool latency cost, step penalty, success bonus).

**User Interaction agent.** A SmolLM3 (Hugging Face, 2025) model conducts a multi-turn dialog with the user to elicit a structured specification of the manufacturing process. The agent prompts for production type, process description, equipment list, sensor placement, and `critical_parameters` (operating ranges and alert thresholds), and emits a JSON object that follows a fixed target schema.

**Data Management agent.** A Struct-SQL (Thaker & Bresler, 2026) model converts the requirements JSON into a relational database schema (entities, columns with types, primary and foreign keys) intended to store telemetry, equipment metadata, and anomaly events for the resulting DT.

**Digital Twin Builder agent.** A Qwen3-Coder-30B (Yang et al., 2025a) model takes the pair ⟨requirements JSON,

DB schema⟩ and generates an executable PyChrono module that instantiates the simulation, registers sensors and IP cameras, and exposes hooks for telemetry persistence into the schema produced by the Data Management agent. We additionally evaluate its outputs offline against a battery of 16 AST and structural tests on a steel-casting/cement-plant benchmark (Section 3.2)

**Complex anomaly-detection agent.** Sensor and video streams are grouped by zone and processed by parallel per-zone detectors —– M2AD (Alnegheimish et al., 2025) on sensors, CKAAD (Fang et al., 2025) on cameras. Stamped events enter a multi-modal correlation block (e.g. a vibration spike with co-occurring smoke); confirmed complex anomalies trigger email alerts. Above-threshold prediction-vs-confirmation discrepancy invokes online fine-tuning. The agent starts unsupervised and adapts to drift. Under distribution shift the agent does not block the pipeline — the M2AD/CKAAD detectors keep running on their pretrained weights and emitting alerts at degraded precision, while confirmed shifted samples incrementally adapt the correlation block.

The pipeline combines **safety mechanisms** that pair orchestrator-side runtime guarantees with evaluation-time gates. At runtime, the orchestrator enforces (i) *bounded execution* via a hard `max_tool_steps` limit, preventing runaway tool-calling loops; (ii) *fail-closed routing* over a fixed tool whitelist (`ipcamera_gige`, `ipcamera_rtsp`, `sensor_manager`, `finish`) — unknown `tool_names` are rejected and surfaced as errors rather than executed; (iii) *tolerance to malformed tool calls* — syntactically invalid `tool_calls` emitted by the LLM are dropped with logging instead of aborting the run or executing a spurious action; and (iv) *explicit termination* via a single `finish` sentinel, so chains end at a controlled point.

## 3. Comparison and Evaluation

### 3.1. Evaluation Methodology

We adapt SimBench (Wang et al., 2024), an LLM-as-a-Judge benchmark for PyChrono-based[1] simulations scoring code on six weighted dimensions out of 100 (Completeness 40, Correctness 30, Quality 10, Efficiency 10, Error 5, Viz 5), and add a J-LLM for manufacturing DT code. We use a multi-turn steel-casting and a single-turn cement-plant scenario.

### 3.2. Experiment 1: Digital Twin Builder Agent

Lacking expert ground-truth code, we replace SimBench's reference-based judge with a *specification-driven* pointwise (and pairwise) judge seeing only requirements and PyChrono API docs; since CodeBLEU (Ren et al., 2020)/ROUGE-LSUM (Lin, 2004) are unsuited for structurally diverse but functionally equivalent code, we add a 16-test AST suite and report code length.

**Metrics.** The *J-LLM Score* applies the SimBench rubric with $-15$ per missing PyChrono component or incorrect API call. The *Structural Test Rate* (STR) is the fraction of 16 AST tests passed. *Code Length* is in tokens.

**Results.** We evaluate Claude 4.6 Sonnet (Anthropic, 2025), DeepSeek-V3.1 Terminus (DeepSeek-AI, 2024), our pipeline (Qwen3-Coder-30B (Yang et al., 2025a)), GPT-5.2 Thinking (OpenAI, 2025), YandexGPT 5.1 Pro 32B (Yandex, 2025), and Gemini 2.5 Flash (Google DeepMind, 2025); judge is Claude 4.6 Sonnet, all models receive identical inputs (Table 1).

| Model | J-LLM | STR | Tokens |
|---|---|---|---|
| Claude 4.6 Sonnet | 65 | 100.0% | 11 673 |
| Qwen3-Coder-30B (ours) | 55 | 100.0% | 3 057 |
| GPT-5.2 Thinking | 40 | 93.8% | 4 556 |
| DeepSeek-V3.1 Terminus | 35 | 100.0% | 7 530 |
| YandexGPT 5.1 Pro (32B) | 30 | 75.0% | 2 168 |
| Gemini 2.5 Flash | 22 | 6.3% | 16 040 |

*Table 1.* Evaluation results of LLMs on the digital twin generation task. J-LLM Score denotes the pointwise judge score (0–100 scale). STR is the Structural Test Rate (fraction of 16 AST-based static analysis tests passed).

Claude 4.6 Sonnet leads at 65, our pipeline follows at 55, GPT-5.2 reaches 40; DeepSeek-V3.1 scores 35 (structurally complete, domain-inaccurate). YandexGPT 5.1 Pro fails four tests (STR $75.0\%$); Gemini 2.5 Flash is longest (16 040 tokens) but barely parses (STR $6.3\%$). Among three $100\%$-STR models, ours is the second-smallest (3 057 tokens) — decomposition yields compact, structurally complete code.

**Analysis.** STR is necessary but not sufficient: Claude, ours, and DeepSeek-V3.1 all reach STR $100\%$ yet score 65/55/35 on J-LLM, and Gemini's longest output collapses on AST parsing. Our 30B pipeline matches the top STR with the second-smallest output and lands within 10 J-LLM points of Claude 4.6 Sonnet, supporting joint J-LLM + STR + length evaluation.

**Ablation: task decomposition.** We compare Qwen3-Coder-30B as a monolithic prompt vs. inside our pipeline, where the orchestrator splits the task into requirements extraction, schema design, and code generation.

Table 2 shows a $5.5\times$ J-LLM gain under decomposition: monolithic Qwen3-Coder-30B scores 10/100 (broken API, missing components); inside our pipeline it reaches 55/100, surpassing DeepSeek-V3.1 (35) and GPT-5.2 (40), and closes the gap to Claude 4.6 Sonnet to within 10 points.

[1] https://projectchrono.org/pychrono/

| Configuration | J-LLM Score |
|---|---|
| Qwen3-Coder-30B, prompt only | 10 |
| Qwen3-Coder-30B, full pipeline (ours) | 55 |

*Table 2.* Ablation: effect of task decomposition on Qwen3-Coder-30B. "Prompt only" is one monolithic prompt; "Full pipeline (ours)" is the multi-agent system with orchestrated sub-tasks.

Orchestration lets smaller specialized models compete on targeted sub-tasks.

**Analysis.** The $45$-point gap ($10 \rightarrow 55$) isolates two effects of decomposition: the code agent receives structured requirements JSON and database schema instead of raw natural language, and operates in a narrow PyChrono-only context. A 30B specialised model inside the pipeline ($55$) closes the gap to Claude 4.6 Sonnet ($65$) to $10$ points and beats GPT-5.2 ($40$) and DeepSeek-V3.1 ($35$) — decomposition, not scale, is the dominant lever.

### 3.3. Human Assessor Evaluation: Multi-Agent vs. Single-Shot Baselines

We complement LLM-as-a-Judge with a blind human evaluation on 118 production scenarios covering discrete assembly, machining, thermal treatment, and food-and-beverage lines. Each scenario specifies stages, equipment, sensors, critical parameters, and expected data flows.

**Conditions.** Per scenario we produce one DT artifact (SQL + Python) per condition. DigitalTwinBuilder (ours): full pipeline. *Terminus*: single-shot DeepSeek-V3.1 returning both artifacts. *DeepSeek-V4-Flash*: analogous single-shot baseline with the much larger model. This yields $118 \times 3$ DT solutions.

We use the following assessment protocol. Domain-aware assessors blindly rated the three anonymized solutions per scenario on 1-5 scales: **Functional Completeness/Usability** and **Code Readability**. Minor syntax issues were ignored.

**Results.** Table 3 reports mean scores over the 118 scenarios.

| Pipeline | Func. Compl. | Readability |
|---|---|---|
| Terminus (single-shot) | 2.75 | 3.08 |
| DigitalTwinBuilder (multi-agent, ours) | 3.18 | 3.41 |
| DeepSeek-V4-Flash (single-shot) | 3.41 | 3.51 |

*Table 3.* Mean human-assessor scores (1–5 scale) over 118 manufacturing scenarios for our multi-agent pipeline (DigitalTwinBuilder) and two single-shot baselines (Terminus (DeepSeek-AI, 2024) and DeepSeek-V4-Flash (DeepSeek-AI, 2026)). Functional Completeness measures the share of stated requirements correctly reflected in the generated SQL and PyChrono code; Readability captures structural clarity, naming, and commenting. Each scenario was rated blindly under all pipeline labels.

**Analysis.** Our pipeline beats Terminus on Functional Completeness (3.18 vs. 2.75), which often omits sensors, threshold checks, or schema-consistent DB writes. DeepSeek-V4-Flash is slightly higher (3.41), but it is a frontier model used end-to-end whereas our 30B coder is orchestrated with smaller components: matching completeness within $\sim 0.23$ using an order-of-magnitude smaller backbone supports our claim that decomposition, not scale, is the main lever.

## 4. Conclusion

We presented a multi-agent system for end-to-end industrial DT construction with a Nemotron-Orchestrator-8B (NVIDIA, 2025) planner routing to specialized agents for schema design, code generation, and anomaly detection. Decomposition is the dominant lever: our 30B pipeline reaches $55/100$ J-LLM at $100\%$ STR ($5.5 times$ over single-LLM ablation), within 10 points of Claude 4.6 Sonnet, and a 118-scenario human study confirms it matches a much larger single-shot frontier model on completeness. Taken together, these results suggest that for safety-critical industrial code generation, a carefully orchestrated mid-sized open model can be a practical alternative to much larger proprietary frontier models, especially when on-premise deployment and auditability matter.

In the future, we plan to add a technical-documentation analysis agent for sensors, as well as integrate auto-prompting algorithms that automatically adapt the prompts of LLM agents to topic shifts in user queries.

## Limitations

Evaluation focuses on assembly-line and machining processes; other domains remain to be validated. Performance depends on the underlying LLMs, with non-determinism and context-window limits. Security concerns (prompt injection, formal safety) are not fully addressed. The pipeline also assumes a reasonably well-specified natural-language description of the process: ambiguous or incomplete user inputs can silently propagate through schema design into generated code, and our human study, while broad in scenario coverage, relies on a small pool of expert assessors and therefore reflects only a limited slice of real industrial review practice.

## Impact Statement

This work automates industrial DT construction, lowering engineering effort for real-time monitoring and predictive analytics. Risks include subtle errors in LLM-generated control code with physical consequences, plus prompt-injection and supply-chain attack surfaces. We position the system as an engineering aid requiring human review and independent

safety verification before closed-loop deployment.

## Acknowledgements

The research was supported by ITMO University, project 625135 "Development of software modules for efficient training and deep learning models application".

## References

Alexopoulos, A., Bompotas, A., Kalogeropoulos, N. R., Kechagias, P., Kalogeras, A. P., and Alexakos, C. A model-based approach to automated digital twin generation in manufacturing. *arXiv preprint arXiv:2511.03742*, 2025.

Alnegheimish, S., He, Z., Reimherr, M., Chandrayan, A., Pradhan, A., and D'Angelo, L. $M^2$AD: Multi-sensor multi-system anomaly detection through global scoring and calibrated thresholding. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 258 of *PMLR*, 2025. arXiv:2504.15225.

Amad, H., Astorga, N., and van der Schaar, M. Continuously updating digital twins using large language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*. PMLR, 2025.

Amazon Web Services. AWS IoT TwinMaker – features. `https://aws.amazon.com/iot-twinmaker/features/`, 2024. Accessed: 2026-03-17.

Ansys. Ansys Twin Builder – create and deploy digital twin models. `https://www.ansys.com/products/digital-twin/ansys-twin-builder`, 2025. Accessed: 2026-03-17.

Anthropic. Claude opus 4 & claude sonnet 4 system card. `https://www.anthropic.com/claude-4-system-card`, 2025. Accessed: 2026-03-17.

Automated Imaging Association. GigE Vision, v2.x specification. Technical report, AIA, 2020. URL `https://www.automate.org/vision/vision-standards/vision-standards-gige-vision`.

AVEVA. Braincube: The leading industrial analytics for process health. `https://www.aveva.com/en/products/braincube-the-leading-industrial-analytics-for-process-health/`, 2025. Accessed: 2026-03-17.

Barat, S., Mulpuru, D., Yadav, A., Thogaru, H., and Kulkarni, V. Constructing enterprise digital twins by augmenting LLMs with MDE. In *Proceedings of the 18th Innovations in Software Engineering Conference (ISEC)*, pp. 11:1–11:11. ACM, 2025. doi: 10.1145/3717383.3717391.

DeepSeek-AI. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

DeepSeek-AI. DeepSeek-V4 flash. `https://api-docs.deepseek.com/news/news260424`, 2026. Released April 24, 2026.

Eclipse Foundation. Eclipse Ditto – open source framework for digital twins in the IoT. `https://eclipse.dev/ditto/`, 2024. Accessed: 2026-03-17.

Fang, Q., Su, Q., Lv, W., Xu, W., and Yu, J. Boosting fine-grained visual anomaly detection with coarse-knowledge-aware adversarial learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 17464–17472, 2025. doi: 10.1609/aaai.v39i16.33816.

Gebreab, S., Musamih, A., Salah, K., Jayaraman, R., and Boscovic, D. Accelerating digital twin development with generative AI: A framework for 3D modeling and data integration. *IEEE Access*, 12:185918–185936, 2024. doi: 10.1109/ACCESS.2024.3514175.

Google DeepMind. Gemini 2.5: Pushing the frontier with advanced reasoning and long context. *arXiv preprint arXiv:2507.06261*, 2025.

Holt, S., Liu, T., and van der Schaar, M. Automatically learning hybrid digital twins of dynamical systems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Hugging Face. Smollm3: Smol, multilingual, long-context reasoner. `https://huggingface.co/blog/smollm3`, 2025. Accessed: 2026-03-17.

Lee, J., Li, J., and Yoon, S. From design to operation: Multi-agent AI for virtual in-situ modeling of digital twins in BIM. *Automation in Construction*, 179:106477, 2025. doi: 10.1016/j.autcon.2025.106477.

Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, 2004. Association for Computational Linguistics.

Lin, Y.-Z., Shi, Q., Yang, Z., Latibari, B. S., Shao, S., Salehi, S., and Satam, P. DDD-GenDT: Dynamic data-driven generative digital twin. *arXiv preprint arXiv:2501.00051*, 2025. Accepted at IEEE Transactions on Artificial Intelligence.

Mandal, S. and O'Connor, N. E. OntoSmartDCU: A multi-agent microservice based automated, dynamic ontology and knowledge graph creation framework from real-time

sensor data for the smart DCU digital twin. In *2024 IEEE 10th World Forum on Internet of Things (WF-IoT)*, pp. 784–789. IEEE, 2024. doi: 10.1109/WF-IoT62078.2024.10811210.

Microsoft. Azure digital twins overview. https://learn.microsoft.com/en-us/azure/digital-twins/overview, 2025. Accessed: 2026-03-17.

Mihai, S., Davis, W., Hung, D. V., Trestian, R., Karamanoğlu, M., Barn, B. S., Prasad, R., Venkataraman, H., and Nguyen, H. X. A digital twin framework for predictive maintenance in industry 4.0. Preprint, Middlesex University, 2021. Available at https://api.semanticscholar.org/CorpusID:233295243.

Nie, Q., Tang, D., Zhu, H., and Sun, H. A multi-agent and internet of things framework of digital twin for optimized manufacturing control. *International Journal of Computer Integrated Manufacturing*, 35(10-11):1205–1226, 2022.

NVIDIA. Llama-nemotron: Efficient reasoning models. *arXiv preprint arXiv:2505.00949*, 2025. URL https://arxiv.org/abs/2505.00949.

ONVIF. ONVIF core specification. Technical report, ONVIF, 2024. URL https://www.onvif.org/specs/core/ONVIF-Core-Specification.pdf.

OpenAI. GPT-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Padgham, L. and Winikoff, M. *Developing Intelligent Agent Systems: A Practical Guide*. John Wiley & Sons, 2004.

Pretel, E., Zhinin-Vera, L., Navarro, E., López-Jaquero, V., and González, P. MAS4DT: A novel proposal for developing digital twins following a multi-agent system approach. *Journal of Systems and Software*, 222:112344, 2025. doi: 10.1016/j.jss.2025.112344.

Preuveneers, D., Joosen, W., and Ilie-Zudor, E. Robust digital twin compositions for industry 4.0 smart manufacturing systems. In *2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW)*, pp. 69–78. IEEE, 2018.

PTC. ThingWorx: Industrial IoT software platform. https://www.ptc.com/en/products/thingworx, 2025. Accessed: 2026-03-17.

Qian, C., Cong, X., Yang, C., Chen, W., Su, Y., Xu, J., Liu, Z., and Sun, M. ChatDev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023.

Reinpold, L. M., Wagner, L. P., Gehlhoff, F., Ramonat, M., Kilthau, M., Gill, M. S., Reif, J. T., Henkel, V., Scholz, L., and Fay, A. Systematic comparison of software agents and digital twins: differences, similarities, and synergies in industrial production. *Journal of Intelligent Manufacturing*, 36(2):765–800, 2025.

Ren, S., Guo, D., Lu, S., Zhou, L., Liu, S., Tang, D., Sundaresan, N., Zhou, M., Blanco, A., and Ma, S. Codebleu: a method for automatic evaluation of code synthesis. *arXiv preprint arXiv:2009.10297*, 2020.

Robles, J., Martín, C., and Díaz, M. OpenTwins: An open-source framework for the development of next-gen compositional digital twins. *Computers in Industry*, 152: 104007, 2023. doi: 10.1016/j.compind.2023.104007.

Schulzrinne, H., Rao, A., and Lanphier, R. Real time streaming protocol (RTSP). Technical Report RFC 2326, IETF, 1998. URL https://www.rfc-editor.org/rfc/rfc2326.

Siemens. The most comprehensive digital twin from Siemens. https://blogs.sw.siemens.com/xcelerator/2025/06/03/the-comprehensive-digital-twin-explained/, 2025. Accessed: 2026-03-17.

Thaker, K. and Bresler, Y. Knowledge distillation with structured chain-of-thought for text-to-sql. In *Proceedings of the 39th Canadian Conference on Artificial Intelligence (Canadian AI 2026)*, 2026. arXiv:2512.17053.

Wang, J., Zhang, H., Unjhawala, H. M., Negrut, P., Wang, S., Slaton, K., Serban, R., Wu, J.-L., and Negrut, D. Simbench: A rule-based multi-turn interaction benchmark for evaluating an llm's ability to generate digital twins. *arXiv preprint arXiv:2408.11987*, 2024.

Wang, Z. and Qin, H. Intelligent industrial production process automatic regulation system based on llm agents. In *2024 5th International Conference on Artificial Intelligence and Electromechanical Automation (AIEA)*, pp. 133–137. IEEE, 2024.

Xia, Y., Shenoy, M., Jazdi, N., and Weyrich, M. Towards autonomous system: flexible modular production system enhanced with large language model agents. In *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8. IEEE, 2023.

Xia, Y., Dittler, D., Jazdi, N., Chen, H., and Weyrich, M. Llm experiments with simulation: Large language model multi-agent system for simulation model parametrization in digital twins. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–4. IEEE, 2024a.

Xia, Y., Jazdi, N., and Weyrich, M. Applying large language models for intelligent industrial automation. *atp magazin*, 66(6–7):62–71, 2024b.

Yandex. YandexGPT 5 pro. `https://yandex.cloud/en/services/yandexgpt`, 2025. Accessed: 2026-03-17.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Yang, L., Luo, S., Cheng, X., and Yu, L. Leveraging large language models for enhanced digital twin modeling: Trends, methods, and challenges. *arXiv preprint arXiv:2503.02167*, 2025b.

Zhao, Z., Tang, D., Liu, C., Wang, L., Zhang, Z., Zhu, H., Chen, K., Nie, Q., and Ji, Y. A large language model-based multi-agent manufacturing system for intelligent shopfloors. *Advanced Engineering Informatics*, 69:103888, 2026. doi: 10.1016/j.aei.2025.103888. arXiv:2405.16887.