BRIC-DT: Building Reliable Interfaces for Composable Digital Twins

Cláudio Gomes
Aarhus University
Aarhus, Denmark

Thomas Wright
Aarhus University
Aarhus, Denmark

Peter Gorm Larsen
Aarhus University
Aarhus, Denmark

Lukas Esterle
Aarhus University
Aarhus, Denmark

Philipp Zech
Universität Innsbruck
Innsbruck, Austria

Ana Cavalcanti
University of York
York, United Kingdom

James Woodcock
Southwest University
Chongqing, China
Aarhus University
Aarhus, Denmark
University of York
York, UK

Bentley Oakes
Polytechnique Montréal
Montreal, Canada

Hans Vangheluwe
Antwerp University
Antwerp, Belgium

Abstract

Digital twins (DTs) increasingly combine models, simulators, estimators, optimizers, and knowledge-management services to monitor and steer adaptive cyber-physical systems. Yet most DTs are still engineered as custom assemblies whose interfaces capture data exchange but not the assumptions and guarantees needed for safe reuse. This vision paper argues that DT engineering should move toward *contract-aware interfaces*: explicit contracts that state not only what data a component exchanges, but also the validity, timing, fidelity, robustness, and uncertainty assumptions governing its trustworthy use. We position BRIC-DT as a contract layer for recurring DT services, outline the gap between current interoperability mechanisms and correctness-preserving composition, and set out a research agenda around explicit correctness properties, reusable composition operators, and correctness-preserving evolution.

CCS Concepts

• **Software and its engineering** → **Formal software verification**; *Software design engineering*; • **Computing methodologies** → *Modeling methodologies.*

Keywords

digital twins, composability, contract-based design, formal methods, model-driven engineering

ACM Reference Format:
Cláudio Gomes, Thomas Wright, Peter Gorm Larsen, Lukas Esterle, Philipp Zech, Ana Cavalcanti, James Woodcock, Bentley Oakes, and Hans Vangheluwe. 2026. BRIC-DT: Building Reliable Interfaces for Composable Digital Twins. In *ACM/IEEE 29th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion 2026), October 04–09, 2026, Málaga, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3837062.3839083

1 Introduction

Physical systems are continually reconfigured: sensors are replaced, operating envelopes shift, parts wear, and control strategies change. Their digital counterparts must co-evolve while remaining valid and trustworthy. Yet today's DTs are often improvised assemblies of models, estimators, simulators, optimizers, and data services linked through syntactic interfaces. These interfaces describe data exchange, but rarely state the behavioral assumptions under which predictions, estimates, optimizations, or control recommendations remain valid.

This paper argues that DT interfaces should be *contract-aware*: they should support reasoning about validity, timing, accuracy, robustness, and safety-relevant behavior, not only data exchange. We propose notations, techniques, and tools for BRIC-DT, Building Reliable Interfaces for Composable Digital Twins. The central claim is simple: **DT interoperability is incomplete unless it captures the assumptions under which component behavior remains trustworthy**.

The novelty is not contract-based design itself [20], but its application to DTs. Compared with general CPS engineering, DT engineering has a recurring component vocabulary: models, simulators, estimators, optimizers, knowledge managers, synchronization services, and decision-support services have recognizable responsibilities across domains [4, 9]. These components are assembled by multidisciplinary stakeholders and must track changing physical counterparts, evidence, calibrations, and replacements at runtime. Contracts therefore provide a practical interface layer: they state when a component may be inserted, trusted, monitored, or retired without invalidating the rest of the DT.

This paper contributes a vision rather than a completed framework. It identifies the gap between current DT composition and the assurance needed for reusable, evolving DTs; outlines a component-and-contract view as a basis for formal composition; and sets out an agenda on correctness properties, composition operators, and correctness-preserving evolution. Existing DT platforms already

manage metadata, runtime streams, co-simulation, orchestration, and monitoring; these mechanisms provide natural places to attach restricted contract patterns for validity frames, latency bounds, uncertainty, and operating envelopes. The difficulty is to make such components compose reliably rather than merely exchange data.

2 Background and Gap

DT research has produced many frameworks, reference architectures, and deployment platforms. Feature models now identify recurring DT capabilities across domains [37], but cohesive support for composition and reuse remains limited [10, 18]. Model composition, validity frames [32], and co-simulation [15] have not fully translated into DT practice, and architecture surveys still report fragmentation [5]. The problem is not a lack of interface mechanisms: the Asset Administration Shell (AAS), for example, is a promising substrate for asset interfaces and submodels [38]. The gap is an engineering approach that connects such interfaces with explicit behavioral conditions for trustworthy composition.

Composability and Modularity. Early DTs were typically monolithic and asset-specific. Ferko et al. [8] found that manufacturing DT architectures often implement only parts of the ISO 23247 reference model, emphasizing data and simulation functions over plug-and-play composition or peer interfaces for twin-to-twin interaction. Muctadir et al. [22] propose graph-based consistency checks across heterogeneous model data, including datatypes, port directions, and mirrored connections. Such checks are necessary, but they do not cover behavioral properties [33]. As a result, many DT systems can exchange data without offering assurance that their combined dynamic behavior satisfies correctness properties.

Platforms such as DTaaS [29] and modular reference architectures help with deployment and communication, but not with whether a composed DT preserves the assumptions under which its components were designed, calibrated, or validated.

Cross-domain feature-model research similarly identifies interoperability, internal communication, modularity, scalability, and composability as recurring DT concerns [37]. Still, these classifications remain largely structural and do not specify the behavioral validity conditions needed for trustworthy composition.

Formal Interfaces. A major barrier to DT composability is ensuring that integrated parts preserve the correctness of the individual parts while establishing new properties of the integration. Beyond data exchange, composition requires alignment of behavioral properties, timing assumptions, uncertainty bounds, and operating constraints. Khedr and Fitzgerald [18] report that formal composition contracts and verification of DT behaviors remain rare. The practice still relies on simulation and testing, with few DT verification frameworks and limited interoperability testing methodology [24].

Formal interface specifications and contract-based design offer a promising foundation. Assume-guarantee and contract-based reasoning build on a long line of work on modular reasoning, including rely-guarantee reasoning and design by contract [3, 16, 20]. Kamburjan et al. [17] advocate formal asset models and ontologies for compositional interaction contracts. Standards such as AAS provide modular interfaces and submodels [38], while OPC UA, MTConnect, and FMI improve data and simulation interoperability [6, 13].

Recent work on reactive Type 2 AAS implementations also shows how design-by-contract can formalize API behavior for testing operation chains and interaction effects [21]. This makes AAS a promising enabling substrate for BRIC-DT, but the existing contracts are not yet formulated around the behavioral assumptions and guarantees of common DT components such as simulators, estimators, optimizers, models, and synchronization services. DT composition therefore also requires correctness: a high-uncertainty decision made by a composed DT can trigger unsafe or suboptimal real-world behavior. This limitation is visible even in structured DT feature models which identify capabilities such as simulation, data analytics, feedback and control, and synchronization, but not the assumptions and guarantees that must hold when such capabilities are connected across components [37].

Model-Driven Engineering. Model-driven engineering (MDE) provides abstractions for managing complex DTs as structured artifacts rather than hand-wired software [36]. MDE supports unified specification across domains [12] and underpins layered or multi-view DT architectures [34, 39]. Tao's five-dimensional model [31] and related layered frameworks [28] organize DTs into modular concerns. Ontologies can further strengthen semantic consistency [24]. However, most frameworks remain case-specific and do not integrate formal verification as a first-class element of DT engineering.

The gap is clear: standards and platforms support connectivity, while formal methods provide assurance principles, but DT engineering lacks an integrated methodology that combines both with model-driven abstraction and industrially usable composition practices.

3 BRIC-DT: A Contract Layer for DT Services

The proposed framework starts from the observation that many DTs are assembled from recurring components: *models* capture system dynamics; *simulators* instantiate models; *state estimators* combine sensor data with predictions; *system identifiers* update models from data; *model couplers* combine models or simulations; *optimizers* guide decisions; and *knowledge managers* capture parameters, histories, errors, and traceability [4, 9, 35].

Each role carries correctness properties concerning validity [25], accuracy, robustness, timing, stability, reachability [33], or role-specific constraints. A state estimator may guarantee bounded error under sensor-noise and model-fidelity assumptions; an optimizer may guarantee configurations within a validated search region; a simulator may require a model valid over the relevant operating envelope. Such properties are rarely visible at interfaces, making reuse deceptively easy.

We use a service-oriented view because many deployed DT platforms expose models, estimators, optimizers, data stores, and synchronization mechanisms as callable services or orchestration nodes [4, 9]. This does not exclude coupled models, layered architectures, multi-agent DTs, or system-of-systems views: all can be represented as compositions whose interaction points carry assumptions and guarantees. The service view is a pragmatic presentation layer for BRIC-DT, not an architectural restriction.

Here, a *service* means a process with inputs, outputs, and internal state, formed by the orchestration of a constellation of components [11]. Figure 1 illustrates this perspective using an incubator

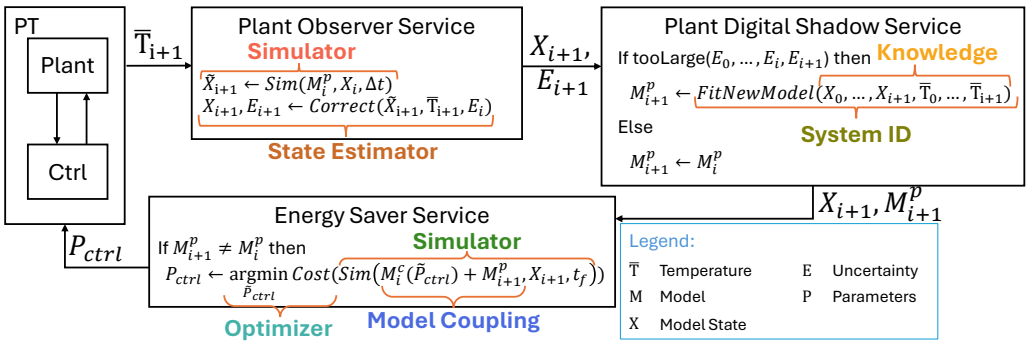


Figure 1: Example DT services built from recurring colored core components.

DT, detailed in [14], that optimizes energy when the plant changes, for example when a mass is added or the lid is slightly opened. In the figure, the `plant observer` service composes a `simulator` and a `state estimator`. The `plant digital shadow` service triggers `system identification` using `plant historical data` stored in the `knowledge manager` when the estimation uncertainty is too high. The resulting `plant model` is then coupled with a `controller model` in the `energy saver` service, which uses an `optimizer` and a `simulator` to find a new optimal controller configuration. If the lid has been opened, the `energy saver` may adjust power output to avoid safety hazards [33]. The component names in Figure 1 correspond to the contract rows in Table 1 and model coupling is treated as the sum operation in Figure 1 that produces a model suitable for simulation.

The vision is that each component exposes an interface contract: assumptions about its inputs and operating context, and guarantees about its outputs and behavior. Specifications may be written in a language such as *signal temporal logic* [19], potentially using specification patterns [7] or constrained fragments that allow efficient reasoning over common DT properties. The contract for an optimizer, for instance, can guarantee validity of new configurations as long as the environment remains within the optimizer's validated search region, using notions related to reach conformance [26, 30] and time-series comparison such as dynamic time warping [27].

Table 1 makes the contract structure concrete. Rows denote recurring DT components from Figure 1, columns denote contract-relevant property dimensions, and each cell is an assume-guarantee fragment: A lists required conditions on inputs, context, or upstream components, while G lists the property provided when those assumptions hold. Reading across a row shows a component's guarantees across fidelity, stability, timing, and robustness; reading down a column shows how a property dimension propagates through the service chain. Contracts compose when an upstream guarantee is strong enough to satisfy a downstream assumption.

Operationally, this means checking whether the guarantees of connected components satisfy the assumptions of their consumers. In Table 1, for example, the model's validity frame and bounded residual satisfy simulator assumptions; the simulator's residual and timing guarantees then support the state estimator's assumptions about model fidelity and measurement cadence. If the estimator's uncertainty becomes too large, the service can trigger system identification, whose resulting model guarantee re-establishes the assumptions needed by the optimizer to search only within a valid region. This style of contract checking is well established [3] and is supported by verification techniques that rely on model checking, theorem proving, runtime verification, and test-case generation. For DTs, the key challenge is not to invent contract theory from scratch, but to adapt it to cyber-physical validity, uncertainty, evolving models, and runtime orchestration. BRIC-DT would use contracts in three lightweight stages: design-time compatibility checks, deployment-time evidence checks using calibration data, validation tests, or benchmark traces, and runtime monitors for live assumptions such as uncertainty, latency, drift, or operating-envelope violations.

4 Research Agenda

Realizing contract-aware DT interfaces suggests three connected research directions. Tool adoption, stakeholder workflows, certification, and platform integration remain outside this short paper.

Direction 1: Make DT correctness properties explicit. A first research direction is to identify the properties that DT interfaces must expose. Some are model-related, such as fidelity [23, 25], robustness, and stability; others are role-specific, such as optimizers respecting constraints and estimators maintaining bounded error. The four columns in Table 1 are illustrative: other dimensions may include relevance, verifiability, substitutability, and faithfulness [25]. A generalized feature taxonomy can help scope which DT capability classes most need formal contracts [37]. A practical route is to encode these properties in a restricted domain-specific language, enabling design-time consistency and compatibility checks and the generation of deployment checks, runtime monitors, and tests.

Direction 2: Treat composition as an engineering operation. A second direction is to move beyond component catalogs toward composition operators, orchestration abstractions, and reusable patterns. Parallel composition, feedback, refinement, additive extension, and integrative merging are natural starting points; MDE can raise them above implementation glue. This need not invent a new orchestration language: existing workflow or co-simulation languages can be execution substrates if their links preserve assumptions, guarantees, and evidence. A central challenge is relating local contracts to system-level requirements, such

Table 1: Example Contract-based properties across the colored core components shown in Figure 1.

Component	Fidelity	Stability	Timing	Robustness
(Coupled) Model	A: Inputs within validity frame G: Output residual bounded by δ_m	A: Bounded inputs G: Bounded outputs; energy conservation	A: $\emptyset$ G: Evaluation of derivatives is $O(n)$	A: Initial conditions bounded by δ_u G: Noise in final state bounded by $\delta_x < \delta_u$
Simulator	A: (Model G) G: Residual $\delta_m + O(\Delta t^4)$ where Δt is step size	A: (Model G) G: At most $O(\Delta t^4)$ energy introduced	A: (Model G) G: Simulation up to T costs $O\left(\frac{T}{\Delta t} \cdot n^3\right)$	A: (Model G) G: Noise in final state bounded by $\delta_x < \delta_u + O(\Delta t^3)$
State Estimator	A: (Simulator G) G: State estimate residual bounded by $\delta_m + O(\Delta t^3)$	A: Observable system; Gaussian noise G: Single estimate, bounded steps	A: (Simulator G) and measurements every Δt G: Estimation completes within Δt	A: Residual $< \delta_m + O(\Delta t^4)$ G: Output residual $< f(\delta_m, \Delta t, Q, R)$ where Q, R are noise co-variances.
System ID	A: Persistently exciting data G: Identified model residual bounded by δ_m	A: BIBO stable model G: Identification terminates in a local optimum.	A: (Model G and Simulator G) G: Identification process costs $O\left(Np^2 \frac{T}{\Delta t} n^3\right)$ for p parameters and N epochs.	A: White noise with covariance R G: Optimum sensitivity $O\left(\frac{1}{\kappa(X)}\right)$ where $\kappa(X)$ is the condition number of data.
Optimizer	A: (Model G) G: Search within validity frame	A: Convex cost and feasible set G: Global optimum, iterations bounded by gradient Lipschitz	A: (System ID G) and epoch takes $\mathcal{T}$ seconds. G: Optimization takes $O\left(\mathcal{T} N p^2 \frac{T}{\Delta t} n^3\right)$	A: Feasible set bounded, closed G: Exploration within feasibility set, solution sensitivity bounded.
Knowledge Manager	A: Sampling jitter and clock offsets are bounded by δ_t G: Queried interpolated data bounded by δ_v	A: Bounded read/write rates G: No data loss or corruption	A: Bounded read/write rates G: Query latency bounded, data can be read after $\Delta\mathcal{T}$ seconds	N/A

as safe operating bounds, energy targets, or decision latency. This also supports emergent-behavior detection: composed-assumption violations, feedback effects, or uncertainty growth can be monitored as properties of the composed DT, not only as component failures.

Direction 3: Preserve correctness as DTs evolve. A third research direction concerns evolution [1, 2]. DTs change because their plants, data sources, and calibrated models change. A composition that was valid yesterday may become invalid after a sensor replacement or operating-envelope shift. Compositional verification offers one route forward: if each component satisfies a contract, the composed DT can be checked against system-level properties through assume-guarantee reasoning, abstraction, and runtime monitoring.

Finally, the vision suggests a practical division of labor: domain engineers describe operating assumptions and validity claims with structured patterns; formal-methods tools discharge proof obligations, generate tests, or synthesize monitors; and DT platforms orchestrate components while preserving and checking contracts.

5 Conclusion

Digital twins are becoming central to *adaptive* cyber-physical systems, but their composition practices lag behind their operational responsibility. The agenda advanced here leads to contract-aware interfaces that expose when a component can be trusted in a composition. This shift changes the role of DT platforms and standards. Connectivity standards remain essential, but they should be complemented by explicit assumptions, guarantees, and evidence stored with the component. A DT marketplace, for example, should expose not only a simulator's data schema but also its operating region, timing assumptions, fidelity claims, and uncertainty bounds.

Reuse is therefore not merely plugging a component into a new DT; it is the **ability to decide, with explicit evidence, whether the component's assumptions hold in the new context**. Evolution likewise becomes a managed engineering activity: when a model is recalibrated, a sensor is replaced, or an optimizer is retuned, contracts indicate which downstream guarantees may need to be rechecked.

By making correctness properties explicit, treating composition as a first-class engineering operation, and preserving contracts as DTs evolve, the community can move toward DTs that are genuinely reusable, adaptable, and dependable.

Acknowledgments

This work is supported by the RoboSAPIENS Project financed by the European Commission's Horizon Europe programme under Grant 101133807, and supported by the Poul Due Jensen Foundation, through the DT-Core project.

References

[1] Tarek Alskaif, Önder Babur, Francis Bordeleau, Loek Cleophas, Benoit Combemale, Joachim Denil, Øystein Haugen, Judith Michael, Phu Nguyen, Tiberiu Seceleanu, et al. 2025. Evolution at the core of digital twin engineering. In *2025 ACM/IEEE 28th International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 210–216.

[2] Nelly Bencomo, Sebastian Götz, and Hui Song. 2019. Models@ run. time: a guided tour of the state of the art and research challenges: N. Bencomo et al. *Software & Systems Modeling* 18, 5 (2019), 3049–3082.

[3] Albert Benveniste, Benoît Caillaud, Dejan Nickovic, Roberto Passerone, Jean-Baptiste Raclet, Philipp Reinkemeier, Alberto Sangiovanni-Vincentelli, Werner Damm, Thomas A. Henzinger, and Kim G. Larsen. 2018. Contracts for System Design. *Foundations and Trends® in Electronic Design Automation* 12, 2-3 (2018), 124–400. doi:10.1561/1000000053

[4] Till Böttjer, Daniella Tola, Fatemeh Kakavandi, Christian R. Wewer, Devarajan Ramanujan, Cláudio Gomes, Peter G. Larsen, and Alexandros Iosifidis. 2023. A review of unit level digital twin applications in the manufacturing industry. *CIRP Journal of Manufacturing Science and Technology* 45 (Oct. 2023), 162–189. doi:10.1016/j.cirpj.2023.06.011

[5] Emilio Carrión and Óscar Pastor. 2023. A Systematic Review of Methodologies for Developing Digital Twins: Insights and Recommendations for Effective Implementation. *PoEM Companion* (2023).

[6] Istvan David, Guodong Shao, Cláudio Gomes, Dawn M. Tilbury, and Bassam Zarkout. 2024. Interoperability of Digital Twins: Challenges, Success Factors, and Future Research Directions. In *Leveraging Applications of Formal Methods, Verification and Validation: Tools and Trends*, Vol. 15223. Springer Nature Switzerland, Crete, Greece, 27–46. doi:10.1007/978-3-031-75390-9_3

[7] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. 1999. Patterns in Property Specifications for Finite-State Verification. In *Proceedings of the 21st International Conference on Software Engineering - ICSE '99*. ACM Press, Los Angeles, California, United States, 411–420. doi:10.1145/302405.302672

[8] Enxhi Ferko, Alessio Bucaioni, Patrizio Pelliccione, and Moris Behnam. 2023. Standardisation in Digital Twin Architectures in Manufacturing. In *2023 IEEE 20th International Conference on Software Architecture (ICSA)*. 70–81. doi:10.1109/ICSA56044.2023.00015

[9] John Fitzgerald, Cláudio Gomes, and Peter Gorm Larsen. 2024. *The engineering of digital twins*. Springer.

[10] Santiago Gil, Peter H. Mikkelsen, Cláudio Gomes, and Peter G. Larsen. 2024. Survey on Open-source Digital Twin Frameworks–A Case Study Approach. *Software: Practice and Experience* (Jan. 2024), spe.3305. doi:10.1002/spe.3305

[11] Santiago Gil, Bentley Oakes, Claudio Gomes, Mirgita Frasheri, and Peter G. Larsen. 2024. Towards a Systematic Reporting Framework for Digital Twins: A Cooperative Robotics Case Study. *SIMULATION* 101, 3 (2024), 313–339. doi:10.1177/00375497241261406

[12] Milapji Singh Gill, Jingxi Zhang, Andreas Wortmann, and Alexander Fay. 2024. Toward Automating the Composition of Digital Twins Within System-of-Systems. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*. 1–4. doi:10.1109/ETFA61755.2024.10710740

[13] Cláudio Gomes. 2019. *Property Preservation in Co-Simulation*. Ph. D. Dissertation. University of Antwerp, Antwerp, Belgium.

[14] Cláudio Gomes, Morten Haahr Kristensen, Mikkel Schmidt Andersen, Prasad Talasila, Hao Feng, Thomas Wright, and Peter Gorm Larsen. 2025. Digital Twin Tutorial: The Incubator Case Study. In *Lecture Notes in Computer Science*. Springer Nature Singapore, Singapore, 68–101. doi:10.1007/978-981-96-4656-2_3

[15] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. 2018. Co-Simulation: A Survey. *Comput. Surveys* 51, 3 (2018), 49:1–49:33. doi:10.1145/3179993

[16] Cliff B. Jones. 1983. Tentative steps toward a development method for interfering programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 5, 4 (1983), 596–619.

[17] Eduard Kamburjan, Vidar Norstein Klungre, Silvia Lizeth Tapia Tarifa, Rudolf Schlatte, Martin Giese, David B Cameron, and Einar Broch Johnsen. 2023. Emerging Challenges in Compositionality and Correctness for Digital Twins. In *CEUR Workshop Proceedings*, Vol. 3507. Technical University of Aachen.

[18] Mennatullah T. Khedr and John S. Fitzgerald. 2025. The Composition of Digital Twins for Systems-of-Systems: A Systematic Literature Review. arXiv:2506.20435 [cs] doi:10.48550/arXiv.2506.20435

[19] Oded Maler and Dejan Nickovic. 2004. Monitoring Temporal Properties of Continuous Signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Yassine Lakhnech, and Sergio Yovine (Eds.). Vol. 3253. Springer Berlin Heidelberg, Berlin, Heidelberg, 152–166. doi:10.1007/978-3-540-30206-3_12

[20] B. Meyer. 1992. Applying 'Design by Contract'. *Computer* 25, 10 (1992), 40–51. doi:10.1109/2.161279

[21] Torben Miny, Sebastian Heppner, Igor Garmaev, Marko Ristin, Björn Otto, Nico Braunisch, Michael Jacoby, Tobias Kleinert, Hans Wernher van de Venn, and Martin Wollschlaeger. 2024. Towards a Test Framework for Reactive (Type 2) Asset Administration Shell Implementations. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*. 01–08. doi:10.1109/ETFA61755.2024.10710957

[22] Hossain Muhammad Muctadir, Eduard Kamburjan, Loek Cleophas, and Mark van den Brand. 2025. A Consistency Management Framework for Digital Twin Models. *SSRN* (Jan. 2025). doi:10.2139/ssrn.5105174

[23] Paula Muñoz. 2022. Measuring the Fidelity of Digital Twin Systems. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*. ACM, Montreal Quebec Canada, 182–188. doi:10.1145/3550356.3558516

[24] Bentley Oakes, Cláudio Gomes, Eduard Kamburjan, Giuseppe Abbiati, Elif Ecem Bas, and Sebastian Engelsgaard. 2024. Towards Ontological Service-Driven Engineering of Digital Twins. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24)*. Association for Computing Machinery, New York, NY, USA, 464–469. doi:10.1145/3652620.3688261

[25] Bentley Oakes, Cláudio Gomes, Peter Gorm Larsen, Joachim Denil, Julien DeAntoni, João Cambeiro, and John Fitzgerald. 2023. Examining Model Qualities and Their Impact on Digital Twins. In *Annual Modelling and Simulation Conference*. Ontario, Canada, 220–232.

[26] Hendrik Roehm, Jens Oehlerking, Matthias Woehrle, and Matthias Althoff. 2016. Reachset Conformance Testing of Hybrid Automata. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*. ACM, Vienna Austria, 277–286. doi:10.1145/2883817.2883828

[27] Pavel Senin. 2008. Dynamic Time Warping Algorithm Review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA* 855, 1-23 (2008), 40.

[28] Sumit Singh, Essam Shehab, Nigel Higgins, Kevin Fowler, Dylan Reynolds, John A Erkoyuncu, and Peter Gadd. 2021. Data Management for Developing Digital Twin Ontology Model. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 235, 14 (Dec. 2021), 2323–2337. doi:10.1177/0954405420978117

[29] Prasad Talasila, Cláudio Gomes, Lars B Vosteen, Hannes Iven, Martin Leucker, Santiago Gil, Peter H Mikkelsen, Eduard Kamburjan, and Peter G Larsen. 2024. Composable Digital Twins on Digital Twin as a Service Platform. *SIMULATION* (Dec. 2024), 00375497241298653. doi:10.1177/00375497241298653

[30] Chencheng Tang and Matthias Althoff. 2024. Efficiently Obtaining Reachset Conformance for the Formal Analysis of Robotic Contact Tasks. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Abu Dhabi, United Arab Emirates, 2305–2311. doi:10.1109/IROS58592.2024.10802263

[31] Fei Tao, He Zhang, Ang Liu, and A. Y. C. Nee. 2019. Digital Twin in Industry: State-of-the-Art. *IEEE Transactions on Industrial Informatics* 15, 4 (April 2019), 2405–2415. doi:10.1109/TII.2018.2873186

[32] Simon Van Mierlo, Bentley Oakes, Bert Van Acker, Raheleh Eslampanah, Joachim Denil, and Hans Vangheluwe. 2020. Exploring Validity Frames in Practice. In *Proceedings of the First International Conference, ICSMM 2020*. Springer, Cham, 131–148. doi:10.1007/978-3-030-58167-1_10

[33] Thomas Wright, Cláudio Gomes, and Jim Woodcock. 2022. Formally Verified Self-adaptation of an Incubator Digital Twin. In *Leveraging Applications of Formal Methods, Verification and Validation. Practice*, Vol. 13704. Springer Nature, Switzerland, 89–109. doi:10.1007/978-3-031-19762-8_7

[34] Chunlong Wu, Youcheng Zhou, Marcus Vinicius Pereia Pessôa, Qingjin Peng, and Runhua Tan. 2021. Conceptual Digital Twin Modeling Based on an Integrated Five-Dimensional Framework and TRIZ Function Model. *Journal of Manufacturing Systems* 58 (Jan. 2021), 79–93. doi:10.1016/j.jmsy.2020.07.006

[35] Honghai Wu, Pengwei Ji, Huahong Ma, and Ling Xing. 2023. A Comprehensive Review of Digital Twin from the Perspective of Total Process: Data, Models, Networks and Applications. *Sensors* 23, 19 (Oct. 2023), 8306. doi:10.3390/s23198306

[36] Philipp Zech, Souvik Barat, Benjamin Nast, Bentley Oakes, Judith Michael, Steffen Zschaler, Balbir Barn, and Ruth Breu. 2026. Model-based digital twin engineering: insights, challenges, and future directions. *Software and Systems Modeling* (04 May 2026). doi:10.1007/s10270-026-01368-8

[37] Philipp Zech, Yanis Mair, Michael Vierhauser, Pablo Oliveira Antonino, Frank Schnicke, and Tony Clark. 2026. A Generalized Feature Model for Digital Twins. arXiv:2603.06308 [cs.SE] https://arxiv.org/abs/2603.06308

[38] Jingxi Zhang, Carsten Ellwein, Malte Heithoff, Judith Michael, and Andreas Wortmann. 2025. Digital Twin and the Asset Administration Shell. *Software and Systems Modeling* 24, 3 (June 2025), 771–793. doi:10.1007/s10270-024-01255-0

[39] Pai Zheng and Abinav Shankar Sivabalan. 2020. A Generic Tri-Model-Based Approach for Product-Level Digital Twin Development in a Smart Manufacturing Environment. *Robotics and Computer-Integrated Manufacturing* 64 (Aug. 2020), 101958. doi:10.1016/j.rcim.2020.101958