MT4DT: Metamorphic Testing for Digital Twins

1st Philipp Zech
Department of Computer Science
University of Innsbruck
Innsbruck, Austria
0000-0002-4952-4337

2nd Sascha Hammes
Unit of Energy Efficient Building
University of Innsbruck
Innsbruck, Austria
0000-0001-5821-5053

3rd Manuel Núñez
Facultad de Informática
Universidad Complutense de Madrid
Madrid, Spain
0000-0001-9808-6401

***Abstract*—Digital Twins (DT) are often exercised in *what-if* scenarios where ground-truth oracles are unavailable. To this end, we present MT4DT, a metamorphic-testing approach for validating DTs under such *what-if* scenarios by checking necessary input–output relations over DT interfaces rather than expected outputs. MT4DT offers a concise procedure embedded in an established test harness to generate follow-up inputs, collect outputs, and verify relations. An illustrative example using a lighting-system DT shows how MT4DT supports practical, oracle-less validation and complements existing simulation and cyber-physical system testing.**

***Index Terms*—Software Testing, Digital Twins, Metamorphic Testing**

I. Introduction

Digital Twins (DTs) enable what-if scenario analysis, predictive maintenance, and optimization across manufacturing, energy, healthcare, and software engineering [1], [2], [3]. However, validating DTs in exploratory scenarios is fundamentally challenging: when testing "what if we add a new sensor?" or "what if usage patterns change?", ground truth oracles are unavailable by definition. Industry frameworks such as DNV-RP-A204 for qualification [4], [5], [6] and TEVV for validation [7], alongside ontology-based credibility guidance [8], [9], provide systematic assurance structures. Yet reviews in manufacturing and systems-of-systems report gaps in comprehensive verification, with predominant reliance on semi-formal approaches [10], [11]. In addition, uncertainty quantification addresses known errors but falls short for emergent scenarios [12], [13], and virtual testbeds offer only limited validation for novel regimes [14], [15].

Complementary to these works, a growing body of research applies metamorphic testing (MT) [16] to simulation models and cyber-physical systems (CPS) where ground truth oracles are scarce. Olsen and Raunak demonstrate that MT systematically increases the validity of simulation models by verifying metamorphic relations (MR) across related runs [17]. Ahlgren et al. apply MT to web-enabled simulation at scale and show robust fault detection in large-scale simulations [18]. In safety-critical domains, MT has validated autonomous-system simulations [19], including scenario-driven approaches for autonomous driving [20]. Beyond correctness, performance-driven MT assesses non-functional behavior of CPS under controlled instrumentation [21], while recent work catalogs reusable MRs [16] to guide systematic MR selection and robustness analysis [22]. Industrial deployments further underscore both opportunity and challenges of applying MT, e.g., Facebook's cyber-cyber and cyber-physical DTs, where rapid evolution, configuration changes, and scale complicate the use of ground truth oracles and motivate automated, relation-driven validation using MT [23].

However, existing MT approaches face two critical limitations when applied to DTs. First, MRs are defined informally without explicit interface contracts: prior work specifies relations over implicit simulation variables or CPS states, but DTs expose typed I/O interfaces with structured payloads that demand formal specifications to distinguish which I/O pairs participate in which relations. Second, MRs usually assume instantaneous or discretized responses [24], which are unsuitable for DTs where physical dynamics introduce settling times, tolerance bands, and continuous trajectories. For instance, asserting monotonicity [22], cf. detecting inverse responses, might require specifying *when* to observe outputs after discrete control inputs and *what tolerance* accounts for continuous physical dynamics [17], [20]. Without explicit formal modeling of DT interface behavior, practitioners cannot systematically derive MRs, localize faults via I/O pairs, or parameterize temporal and numeric tolerances.

We propose **MT4DT**, a metamorphic testing approach for DTs. MT4DT leverages Hybrid Input-Output Automata (HIOA) [25] for formal abstraction of DT interfaces to enable guided MR specification with precise semantics. HIOA are ideal to model DTs as, by integrating differential equations to model continuous state evolution with discrete transitions, they can accurately represent real-time synchronization and data exchange between the physical entity and its digital replica. In addition, MT4DT introduces tolerance- and sampling-aware MRs, specifically designed for accounting for physical dynamics during DT validation. Unlike hybrid simulations where the combination of discrete and continuous dynamics might trigger rollback mechanisms or cause numerical instabilities [26], [27], [28], DTs demand seamless integration of discrete and continuous dynamics. HIOA provide the mathematical foundation to specify MRs spanning both modalities. Finally, we present a structured testing workflow with a feedback loop to trace I/O violations to implementation faults.

We demonstrate MT4DT through an illustrative example: a building lighting-system DT to validate sensor and actuator integration scenarios without ground truth oracles. This work provides actionable insights for software engineers and testers (skills for oracle-less DT validation), system architects and QA leaders (DT-specific testing processes), and decision-makers (systematic testing strategies for DT engineering and

This research has received funding from the Austrian Research Promotion Agency (FFG) under Grant Agreement No.: 927874 (ELEVATE), the State Research Agency (AEI) of the Spanish Ministry of Science and Innovation under grant PID2021-122215NB-C31 (AwESOMe), and the Region of Madrid under grant TEC-2024/COM-235 (DESAFíO-CM).

operation).

II. Metamorphic Testing and Hybrid I/O Automata

MT addresses the oracle problem by exploiting metamorphic relations (MRs) that describe necessary properties of the system under test (SUT) [29], [30]. An MR is defined by a tuple (I, I^f, O, O^f, MR), where I denotes source inputs, I^f denotes follow-up inputs derived from I, O and O^f are the corresponding outputs, and MR specifies the expected relation between them. For example, a monotonicity MR might require that if $I < I^f$, then $O \leq O^f$. Crucially, MRs may relate multiple inputs and their follow-ups simultaneously (cf. conservation laws that involve several sources). MT detects faults by observing MR violations without requiring explicit expected outputs for individual test cases.

Hybrid I/O Automata (HIOA) model systems with both discrete and continuous dynamics [25]. A HIOA distinguishes input variables (U), internal state (X), and output variables (Y), and captures behavior through discrete transitions (D) that change state instantaneously and continuous trajectories (W) that govern variable evolution between transitions. For DT validation, HIOA provide (i) typed I/O interfaces that map to DT communication endpoints, (ii) a unified representation of discrete commands alongside continuous physical responses, and (iii) formal semantics for specifying when to observe outputs via trajectories and what tolerance to apply [25], [31]. This makes them well-suited for DT abstraction where discrete commands trigger continuous responses.

III. MT4DT: Metamorphic Testing for DTs

We now propose MT4DT (cf. Figure 1), a structured testing methodology for DTs that leverages MT to enable validation in the absence of ground truth oracles.

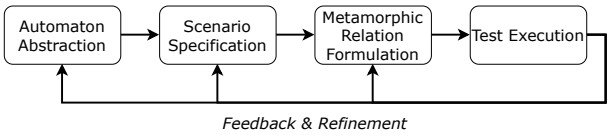

Fig. 1. MT4DT presents a workflow for MT of DTs by mapping DT interfaces to formal contracts, deriving MRs, and tracing violations to system faults.

DTs differ from general hybrid systems through bidirectional physical-digital synchronization, continuous state evolution under real-time constraints, and sensor-actuator coupling with inherent noise and settling dynamics. These characteristics necessitate tolerance-aware MRs that account for physical behavior rather than purely computational logic. While HIOA provide the formal substrate, MT4DT's contribution lies in parameterizing MRs with physical dynamics specific to DT validation.

1) Automaton Abstraction: The DT is modeled as a black-box HIOA, exposing only I/O types (U, Y) while abstracting internal states ($X = \emptyset$). This formalization establishes precise contracts for input parameters and output observations for reasoning about *when* to observe outputs via trajectories (W) and *what tolerance* to apply. Crucially, HIOA's hybrid semantics allow MRs to account for physical dynamics and measurement noise, which are inexpressible in classical black-box MT frameworks that treat systems as instantaneous input-output functions [24].

2) Scenario Specification: Unlike oracle-based testing which requires pre-computed expected outputs for every test scenario, MT4DT scenarios exploit *invariant properties*, e.g., monotonicity, conservation, or invariance [22] that must hold regardless of absolute values. Scenario selection thus ideally prioritizes operational regimes with high uncertainty or novelty, cf. *what-if* scenarios, to maximize the potential to uncover hidden faults.

3) Metamorphic Relation Formulation: MRs specify necessary properties over I/O pairs without requiring expected outputs. MR identification starts from the DT interface specification, viz. its inputs, outputs, and physical domain. Practitioners analyze system properties to match physics with MR patterns, e.g., by checking if inputs monotonically affect outputs, identifying conserved quantities (e.g., energy), and assessing whether steady states stabilize. For DTs exhibiting continuous physical dynamics, three MR patterns are frequently applicable [22], [29]:

- *Monotonicity*: If the input increases ($I < I^f$), then the output should not decrease ($O^f - \epsilon_{\text{noise}} \geq O$). Monotonicity allows to detect inverse responses or scaling errors.
- *Conservation*: Redistributing inputs while preserving totals ($\sum I_j = \sum I_j^f$) should maintain output totals ($|\sum O_j - \sum O_j^f| \leq n \cdot \epsilon_{\text{noise}}$). Conservation allows to validate multi-component coupling.
- *Invariance*: Repeating inputs ($I = I^f$) should yield consistent outputs ($|O - O^f| \leq \epsilon_{\text{repeat}}$). Invariance allows to detect non-determinism or state corruption.

These relations enable systematic validation without ground truth oracles. However, not all patterns apply to all DT domains. For instance, monotonicity is irrelevant for systems with oscillatory or hysteretic behavior. Thus, practitioners should select MRs based on domain-specific properties, as the rigor and relevance of these relations are critical to testing validity and fault-detection capability [29].

a) Temporal and Tolerance Parameters: Temporal and tolerance parameters must account for physical dynamics. The settling time (Δt_{settle}) ensures outputs stabilize before measurement, and the noise tolerance (ϵ_{noise}) captures sensor uncertainty. These parameters are determined through (i) analytical bounds from component specifications (e.g., actuator datasheets), (ii) empirical calibration under controlled conditions, and (iii) safety margins for robustness. For multi-component conservation relations, tolerances accumulate linearly ($(\epsilon_{\text{noise}})$) under the assumption of uncorrelated sensor noise. This conservative bound is valid when measurement errors are independent, but correlated errors require adjusted bounds.

4) Test Execution: Original inputs (I) and systematically transformed follow-ups (I^f) are submitted to the DT's I/O interfaces. Outputs (O, O^f) are sampled at intervals defined by Δt_{settle} and checked against MRs. Violations flag potential faults, e.g., a monotonicity violation ($O^f + \epsilon_{\text{noise}} < O$ despite $I < I^f$) might indicate that the DT implementation deviates from its HIOA specification and reveal faults in either the computational logic or component integration.

Follow-up inputs I^f are generated by systematically transforming I according to the selected MR. For monotonicity, I^f is incremented by a fixed delta or percentage; for conservation,

I^f redistributes values across components by preserving the total; for invariance, I^f is identical to I. The transformation strategy depends on input domain and the MR's specific semantics.

5) Feedback & Refinement: Violations are analyzed to localize faults. The HIOA abstraction enables *systematic traceability*. MRs are defined over explicit I/O pairs (U, Y), so that they pinpoint to *what* failed rather than opaque internal states. Root-cause analysis proceeds by (i) replaying violated inputs with instrumentation to capture intermediate trajectories, (ii) comparing observed settling times against Δt_{settle} to detect timing issues, and (iii) validating tolerance bounds via repeated trials to distinguish faults from noise. Fixes are versioned alongside the HIOA specification, and MRs are refined as system understanding improves, e.g., tightening ϵ_{noise} after hardware (re-)calibration.

In synopsis, MT4DT provides a systematic testing workflow for DT validation under lacking ground truth oracles. By formalizing I/O contracts and parameterizing MRs with physical dynamics, MT4DT extends DT validation to exploratory *what-if* scenarios where oracle-based testing fails.

IV. Illustrative Case Study

To illustrate the MT4DT approach, we consider our recent work where we present a DT for building operations [32]. The setting is an office "Living Lab" where an DT maintains a synchronized virtual representation of the lighting system for control and monitoring of physical luminaires and sensors via a unified user interface. This domain exhibits characteristic challenges for DT validation: (i) *continuous physical dynamics* where illuminance evolves over time following set-point commands and external factors (e.g., daylight), (ii) *sensor noise and actuator variance* where repeated commands might yield slightly different readings, and (iii) *emergent interactions* when adding luminaires or sensors which affect system dynamics in ways unpredictable from individual component specifications. Collectively, these render ground truth oracle construction infeasible.

1) Automaton Abstraction: The DT is abstracted as a black-box HIOA with:

- $U = \{\text{set-point commands}\}$: discrete input actions
- $Y = \{\text{illuminance readings}\}$: continuous output variables sampled at intervals
- W: trajectories modeling physical illuminance evolution between discrete sampling instants

Crucially, the HIOA enables explicit reasoning about *when* to observe outputs (via trajectory W and sampling intervals Δt_{settle}) and *what tolerance* to apply (measurement noise bounds).

2) Scenario Specification: We examine integration of a new luminaire and sensor into the existing lighting system. This configuration change forces the DT to dynamically recompute illumination contributions and recalibrate spatial light distribution models. Unlike orcale-based testing which would require pre-computed expected illuminance values for every spatial point, daylight condition, and lamp configuration, MT4DT exploits invariant properties (monotonicity, conservation, invariance) that must hold *regardless* of absolute output values. The scenario's complexity stems from unpredictable daylight interference through windows, spatial coupling where one luminaire's output affects sensors near adjacent luminaires, and non-instantaneous lamp responses.

An input I (following our previous work [32]) is a triple $I = (T, P, V)$, where:

- T is the command topic (e.g., "set-point").
- P is the device path (e.g., /device/lum1).
- V is the new target illuminance, e.g., $V = 500\,\text{lx}$.

An output O is a pair $O = (T, V)$, where:

- T is the response topic (e.g., "illuminance").
- V is the measured illuminance value, e.g., $V = 600\,\text{lx}$

3) Metamorphic Relation Specification: We selected three MRs based on the smart room interface and physical properties. MR1 (monotonicity) exploits the linear relationship between lamp power and illumination. MR2 (conservation) leverages total light output constancy for fixed aggregate power. MR3 (invariance) validates temporal stability under constant conditions. These MRs were chosen because they (i) exercise all interface operations, (ii) reflect physical lighting principles, and (iii) detect common faults (calibration errors, actuator failures, timing issues).

a) MR1: Monotonicity (Temporal settling and noise tolerance): Increasing the set-point should not decrease observed illuminance, while accounting for physical settling time and sensor noise. Formally, given inputs $I = (T, P, V)$ and $I^f = (T, P, V^f)$ with outputs $O = (T, {}^tV_o)$ and $O^f = (T, {}^tV_o^f)$:

$$V < V^f \implies {}^tV_o^f - \epsilon_{\text{noise}} \geq {}^tV_o$$

Temporal constraint: Outputs are sampled at time $t = t_{\text{cmd}} + \Delta t_{\text{settle}}$ after corresponding command issuance, where $\Delta t_{\text{settle}} = 600\,\text{ms}$ (empirically determined as $1.2\times$ maximum lamp ramp time).
Tolerance: $\epsilon_{\text{noise}} = 5\,\text{lx}$ accounts for sensor quantization and ambient fluctuation.

This MR prevents false violations during transient dimming.

b) MR2: Conservation (Multi-luminaire stability): Redistributing set-points across n luminaires while preserving total target illuminance should maintain total measured output within system-wide tolerances. For luminaires $I_j = (T, P_j, V_j)$ $(1 \leq j \leq n)$ and follow-ups $I_j^f = (T, P_j, V_j^f)$ with outputs $O_j = (T, V_{o_j})$ and $O_j^f = (T, V_{o_j}^f)$:

$$\sum_{j=1}^{n} V_j = \sum_{j=1}^{n} V_j^f \implies \left| \sum_{j=1}^{n} {}^tV_{o_j} - \sum_{j=1}^{n} {}^tV_{o_j}^f \right| \leq n \cdot \epsilon_{\text{noise}}$$

Temporal constraint: All outputs are sampled simultaneously at $t = \max_j(t_{\text{cmd},j}) + \Delta t_{\text{settle}}$ (after corresponding command issuance) to ensure system-wide steady state.
Tolerance: $\epsilon_{\text{noise}} = 5\,\text{lx}$ accounts for sensor quantization and ambient fluctuation.

This multi-input MR validates spatial coupling, a property impossible to specify without relating multiple luminaire behaviors.

c) MR3: Invariance (Repeatability under noise): Repeating identical set-points should yield consistent illuminance within measurement uncertainty. For inputs $I = (T, P, V)$ and $I^f = (T, P, V^f)$ with $V = V^f$ and outputs $O = (T, {}^tV_o)$, $O^f = (T, {}^tV_o^f)$:

$$V = V^f \implies |{}^tV_o - {}^tV_o^f| \leq \epsilon_{\text{repeat}}$$

Temporal constraint: Second command issued $\geq 2\,\mathrm{s}$ after first to avoid caching artifacts in the DT with outputs sampled at $t = t_{\text{cmd}} + \Delta t_{\text{settle}}$ after corresponding command issuance. **Tolerance**: $\epsilon_{\text{repeat}} = 5\,\mathrm{lx}$ allows for sensor drift between measurements.

This relation detects state corruption or non-deterministic faults (e.g., race conditions in lamp driver updates). Settling times for the above relations (cf., M1, M2, and M3) were determined from lamp manufacturer ramp specifications ($450\,\mathrm{ms}$ nominal) with $1.3\times$ safety margin. Noise tolerance was measured empirically over 50 stable-state readings that yielded $\sigma \approx 2.1\,\mathrm{lux}$. Accordingly, we use $2.5\sigma \approx 5\,\mathrm{lux}$.

Fault Detection Example. During testing, MR1 detected a violation: issuing $I = (\text{“set”}, \text{“/lum3”}, 600\,\mathrm{lx})$ followed by $I^f = (\text{“set”}, \text{“/lum3”}, 800\,\mathrm{lx})$ yielded ${}^tV_o = 612\,\mathrm{lx}$ but ${}^tV_o^f = 580\,\mathrm{lx}$, a $32\,\mathrm{lx}$ *decrease* despite a higher set-point (violating ${}^tV_o^f - 10 \geq {}^tV_o$). Root-cause analysis traces this to a scaling error in the DT's photometric model: the newly added luminaire's contribution was multiplied by 0.8 instead of 1.0 due to a copy-paste bug. *Classical testing would have missed this*: Without knowing the "correct" illuminance values (dependent on room geometry, reflectance, daylight, etc.), a ground truth oracle cannot flag $580\,\mathrm{lx}$ as wrong. MT4DT flagged the *relationship violation*, thus revealing the fault without requiring absolute ground truth.

Interpretation and Use. These relations are systematically exercised by sending original and transformed inputs via the DT's interface [32] and comparing outputs sampled at specified intervals. Violations reveal a plausible DT fault, without requiring a ground truth oracle for every scenario. Subsequently, relevant changes can be applied to the DT to resolve the fault (cf. *Feedback & Refinement*). Any refinements should be versioned alongside the DT's HIOA specifcation.

V. Discussion

MT4DT advances DT validation by directly addressing the ground truth oracle problem inherent in *what-if* analyses. By verifying I/O relations rather than absolute expected values, MT provides a systematic approach to validating DTs under emergent behaviors and inherent uncertainty.

MT4DT's use of HIOA addresses specific limitations of existing MT approaches for DTs. First, HIOA provide explicit interface contracts that map to DT communication endpoints to enable systematic identification of which I/O pairs participate in MRs. Second, HIOA trajectories capture continuous dynamics and enable MRs to specify *when* to observe outputs and *what tolerance* accounts for noise. Third, MRs over explicit interfaces pinpoint specific I/O pairs rather than opaque failures and enable targeted root-cause analysis via replay, settling time comparison, and noise trials. The illustrative example (cf. Section IV) demonstrates how a monotonicity violation reveals a scaling bug that oracle-based testing would have missed. This black-box perspective is intentional as MT4DT validates *externally observable behavior* and complements white-box techniques [31], [12] without requiring internal state exposure, which is often infeasible for proprietary DTs.

However, MR identification for now remains domain-dependent. While our three patterns apply broadly to continuous-state DTs, practitioners must assess domain fit. Future work should thus develop systematic catalogs that link DT interface patterns to applicable MRs.

A. Benefits of Using MT4DT

Oracle-Free Validation. MT4DT enables systematic testing in exploratory and emergent (cf. *what-if*) scenarios where ground truth oracles are absent, and thus addresses a critical gap identified in manufacturing [10] and systems-of-systems engineering [11].

Proactive Fault Detection. MT4DT identifies emergent faults and systemic inconsistencies that arise from complex DT interactions (e.g., spatial coupling in multi-luminaire systems) that conventional, ground truth oracles overlook.

Intrinsic Goal Alignment. By verifying fundamental properties like conservation, MT4DT inherently contributes to assessing DTs' performance against sustainability and efficiency objectives [33].

Enhanced Traceability. HIOA abstractions capture system behavior at I/O interfaces. Violating an MR thus allows tracing directly to those implementation components that deviate from their specifications.

Formal Reproducibility. The mathematical rigor of HIOA and MRs ensures transparent and reproducible testing across diverse deployment contexts, and complements credibility frameworks like DNV-RP-A204 [4] and TEVV [7].

B. Challenges and Limitations

Manual Relation Formalization. Developing rigorous, domain-specific MRs for complex DTs demands significant expertise [29]. Future work on pattern catalogs [22] can reduce this effort (cf. Section VI).

Comprehensive Scenario Coverage. Efficiently generating test cases to explore vast configuration spaces remains non-trivial, necessitating advanced scenario generation [34].

Granular Root Cause Analysis. While HIOA aid fault localization, pinpointing specific root causes within multi-component architectures from MR violations requires sophisticated diagnostics [35].

Scalability for Advanced Deployments. Adapting MT4DT for real-time, distributed, or safety-critical applications, along with CI/CD integration, requires dedicated engineering [36].

Beyond I/O Relations. Future work must leverage HIOA's full expressiveness to formulate advanced MRs probing hybrid dynamics (e.g., phase-space invariants, Lyapunov stability properties) beyond current I/O-centric patterns [31], [37].

VI. Future Plans

Our immediate goal is to develop a working prototype to demonstrate the practical feasibility of MT4DT. Following this, we plan to (i) develop a pattern catalog of reusable MR templates parameterized by settling time, tolerance bands, and sampling strategies; (ii) automate MR derivation from interface specifications to reduce manual effort; (iii) extend the prototype to support continuous and distributed testing in dynamic environments; (iv) advance diagnostic techniques for actionable root-cause analysis; (v) devise training and collaborative best practices to facilitate broader adoption; and (vi) pursue quantitative assessment using DevOps KPIs such as MTTD or DER [38]. These systematic efforts are paramount for fully integrating the MT4DT methodology into robust, modern software engineering processes for DTs.

References

[1] F. Tao, H. Zhang, A. Liu, and A. Y. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on industrial informatics*, vol. 15, no. 4, pp. 2405–2415, 2018.

[2] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: enabling technologies, challenges and open research," *IEEE access*, vol. 8, pp. 108 952–108 971, 2020.

[3] R. Kimmel, J. Michael, A. Wortmann, and J. Zhang, "Digital Twins for Software Engineering Processes," in *ICSE '25: ACM/IEEE 47th International Conference on Software Engineering. Companion Proceedings*. ACM, 2025, preprint.

[4] K. Eriksson and C. Markussen, "Quality Assurance of Digital Twins," in *International Conference on Offshore Mechanics and Arctic Engineering*, vol. 86830. American Society of Mechanical Engineers, 2023, p. V001T01A020.

[5] C. Bell, M. Celnik, J. Devgun, O. Hansen, W. Osborn, and G. Faiz, "Providing Assurance of Digital Twins," in *SPE Offshore Europe Conference and Exhibition*. SPE, 2023, p. D021S006R001.

[6] O. H. Hansen and V. Jaiswal, "A Framework for Trustworthy Digital Twins Over their Life Cycle," in *Offshore Technology Conference Brasil*. OTC, 2023, p. D021S024R001.

[7] G. Waters, "Testing, Evaluation, Verification and Validation (TEVV) of Digital Twins: A Comprehensive Framework," *arXiv preprint arXiv:2507.04555*, 2025.

[8] E. Vanderhorn and S. Valluri, "Guidance on the Verification and Validation of Digital tTwins," in *SNAME Offshore Symposium*. SNAME, 2024, p. D011S004R001.

[9] E. VanDerHorn, B. Corbett, and A. Costa, "Establishing Credibility: Verification and Validation of Digital Twins," in *Offshore Technology Conference*. OTC, 2025, p. D031S034R002.

[10] J. Bitencourt, A. Wooley, and G. Harris, "Verification and validation of digital twins: a systematic literature review for manufacturing applications," *International Journal of Production Research*, vol. 63, no. 1, pp. 342–370, 2025.

[11] M. T. Khedr and J. S. Fitzgerald, "The Composition of Digital Twins for Systems-of-Systems: a Systematic Literature Review," *arXiv preprint arXiv:2506.20435*, 2025.

[12] K. Sel, A. Hawkins-Daarud, A. Chaudhuri, D. Osman, A. Bahai, D. Paydarfar, K. Willcox, C. Chung, and R. Jafari, "Survey and perspective on verification, validation, and uncertainty quantification of digital twins for precision medicine," *Digital Medicine*, vol. 8, no. 1, p. 40, 2025.

[13] G. Maculotti, M. Marschall, G. Kok, B. A. Chekh, M. van Dijk, J. Flores, G. Genta, P. Puerto, M. Galetto, and S. Schmelter, "A Shared Metrological Framework for Trustworthy Virtual Experiments and Digital Twins," *Metrology*, vol. 4, no. 3, pp. 337–363, 2024.

[14] U. Dahmen, "Verification and Validation of Digital Twins and Virtual Testbeds," *International Journal of Advances in Applied Sciences*, vol. 11, no. 1, pp. 47–64, 2022.

[15] E. Y. Hua, S. Lazarova-Molnar, and D. P. Francis, "Validation of Digital Twins: Challenges and Opportunities," in *2022 Winter Simulation Conference (WSC)*. IEEE, 2022, pp. 2900–2911.

[16] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: a new approach for generating next test cases," Department of Computer Science, Hong Kong University of Science and Technology, Tech. Rep. HKUST-CS98-01, 1998.

[17] M. Olsen and M. S. Raunak, "Increasing validity of simulation models through metamorphic testing," *IEEE Transactions on Reliability*, vol. 68, no. 1, pp. 1–18, mar 2019.

[18] J. Ahlgren, M. E. Berezin, K. Bojarczuk, E. Dulskyte, I. Dvortsova, J. George, N. Gucevska, M. Harman, M. Lomeli, E. Meijer, S. Sapora, and J. Spahr-Summers, "Testing web enabled simulation at scale using metamorphic testing," in *Proceedings of the 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '21)*. IEEE Press, 2021, pp. 140–149.

[19] J. G. Adigun, L. Eisele, and M. Felderer, "Metamorphic testing in autonomous system simulations," in *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2022, pp. 330–337.

[20] Y. Zhang, D. Towey, M. Pike, J. C. Han, Z. Q. Zhou, C. Yin, Q. Wang, and C. Xie, "Scenario-driven metamorphic testing for autonomous driving simulators," *Software Testing, Verification and Reliability*, vol. 34, p. e1892, 2024.

[21] J. Ayerdi, P. Valle, S. Segura, A. Arrieta, G. Sagardui, and M. Arratibel, "Performance-driven metamorphic testing of cyber-physical systems," *IEEE Transactions on Reliability*, vol. 72, no. 2, pp. 827–845, 2023.

[22] Z. Ying, D. Towey, A. Bellotti, C. Chua, and Z. Q. Zhou, "Metamorphic relation patterns for metamorphic testing, exploration and robustness," *Software Testing, Verification and Reliability*, vol. 35, p. e70003, 2025.

[23] J. Ahlgren, K. Bojarczuk, S. Drossopoulou, I. Dvortsova, J. George, N. Gucevska, M. Harman, M. Lomeli, S. M. M. Lucas, E. Meijer, S. Omohundro, R. Rojas, S. Sapora, and N. Zhou, "Facebook's cyber–cyber and cyber–physical digital twins," in *Proceedings of the 25th International Conference on Evaluation and Assessment in Software Engineering (EASE '21)*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 1–9.

[24] S. Segura, D. Towey, Z. Q. Zhou, and T. Y. Chen, "Metamorphic Testing: Testing the Untestable," *IEEE Software*, vol. 37, no. 3, pp. 46–53, 2018.

[25] N. A. Lynch, R. Segala, and F. W. Vaandrager, "Hybrid I/O automata," *Information and Computation*, vol. 185, no. 1, pp. 105–157, 2003.

[26] G. P. Fang, "An Efficient Method to Simulate Threshold-crossing Events," in *2008 IEEE International Behavioral Modeling and Simulation Workshop*. IEEE, 2008, pp. 96–99.

[27] A. Falcone, A. Garro, M. S. Mukhametzhanov, and Y. D. Sergeyev, "Simulation of Hybrid Systems Under Zeno Behavior Using Numerical infinitesimals," *Communications in Nonlinear Science and Numerical Simulation*, vol. 111, p. 106443, 2022.

[28] T. Park and P. I. Barton, "State Event Location in Differential-algebraic Models," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 6, no. 2, pp. 137–165, 1996.

[29] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. Tse, and Z. Q. Zhou, "Metamorphic Testing: A Review of Challenges and Opportunities," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–27, 2018.

[30] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortés, "A survey on metamorphic testing," *IEEE Transactions on software engineering*, vol. 42, no. 9, pp. 805–824, 2016.

[31] M. Barth and M.-C. Jakobs, "Test-case generation with automata-based software model checking," in *International Symposium on Model Checking Software*. Springer, 2024, pp. 248–267.

[32] P. Zech, S. Senoner, E. Goldin, C. Zallinger, S. Hammes, and J. Michael, "Model-driven Digital Twins for AECO," in *28th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 2025, accepted for publication.

[33] I. David, "SusDevOps: Promoting Sustainability to a First Principle in Software Delivery," in *ICSE '25: ACM/IEEE 47th International Conference on Software Engineering. Companion Proceedings*. ACM, 2025, preprint.

[34] S. Wynn-Williams, R. Tyrrell, V. Pantelic, M. Lawford, C. Menghi, P. Nalla, and H. Artail, "Can Generative AI Produce Test Cases? An Experience from the Automotive Domain," in *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, 2025, pp. 456–467.

[35] K. Zhang, I. Shpitser, S. Magliacane, D. Bacciu, F. Wu, C. Zhang, and P. Spirtes, "IEEE Transactions on Neural Networks and Learning Systems Special Issue on Causal Discovery and Causality-Inspired Machine Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 4899–4901, 2024.

[36] J. G. Süß, S. Swift, and E. Escott, "Using DevOps toolchains in Agile model-driven engineering," *Software and Systems Modeling*, vol. 21, no. 4, pp. 1495–1510, 2022.

[37] Z. Sadri-Moshkenani, J. Bradley, and G. Rothermel, "Test Case Generation and Test Oracle Support for Testing CPSs using Hybrid Models," *arXiv preprint arXiv:2309.07994*, 2023.

[38] R. Amaro, R. Pereira, and M. Mira da Silva, "DevOps metrics and KPIs: a multivocal literature review," *ACM Computing Surveys*, vol. 56, no. 9, pp. 1–41, 2024.