Toward a Unified Method for Digital Twin Development: A Systematic Comparison of Existing Frameworks and Process Models

Anton Zagzin

Vilnius University, Faculty of Mathematics and Informatics,
Naugarduko g. 24, LT-03225 Vilnius, Lithuania
anton.zagzin@mif.stud.vu.lt

Abstract. The rapid growth of interest in Digital Twins has produced a heterogeneous landscape of development methods, frameworks and process models. Existing reviews describe the state of the art in general terms, focus on selected dimensions such as methodologies or Model-Driven Engineering techniques, or propose domain-specific engineering approaches, but no unified view systematically compares these contributions and distils a reference method adaptable across domains. This paper reports the first stage of my research toward such a unified method by systematically comparing existing frameworks and process models and synthesising the result into a coherent reference method. The methodology combines systematic literature review, comparative method analysis and inductive synthesis. The paper presents the current progress, the proposed contribution and open issues.

Keywords: digital twin, development method, process model, systematic comparison, model-driven engineering, reference method.

1 Introduction

Digitalisation and the increasing integration of software into physical systems are reshaping how complex products and infrastructures are designed, operated and maintained. In this context, the concept of the Digital Twin (DT)—a virtual representation of a physical system kept continuously aligned with its real-world counterpart—has become a central enabler of data-driven decision-making and lifecycle optimisation [1, 2]. Digital Twins are now explored and deployed in manufacturing, transportation, energy systems, healthcare and smart cities, with promises of improved monitoring, predictive maintenance, virtual experimentation and new servitisation-oriented business models [3, 7].

Copyright © Anton Zagzin, 2026. Published by Vilnius University Press. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (CC BY), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.
DOI: https://doi.org/10.15388/LMITT.2026.34

The rapid growth of interest has led to an explosion of heterogeneous proposals for how DTs should be conceptualised, architected and engineered. Existing work emphasises that Digital Twins are inherently interdisciplinary artefacts that combine models of physical behaviour, software components, data management infrastructures and domain knowledge [7]. As a result, the development of a Digital Twin is no longer a purely technical programming task, but a complex engineering endeavour that spans requirements engineering, domain modelling, system architecture, data and simulation models, integration with legacy systems, and runtime operations.

Because of this complexity, Digital Twins cannot be engineered in an ad-hoc manner. Systematic development methods, frameworks and process models are needed to guide practitioners through the main activities and decisions involved in building and evolving DTs. Several strands of literature address this methodological dimension: dedicated reviews of DT methodologies [6, 7], broader reviews that categorise DT development alongside Cyber-Physical Systems and Product-Service Systems [3], mapping studies of MDE techniques and process-model comparisons in the DT context [8, 9], and concrete frameworks for model-driven and ontology-based DT engineering [10, 11].

Taken together, the existing literature highlights an important gap. On the one hand, several surveys describe the state of the art in general or focus on particular dimensions such as methodologies [6], process models [9] or MDE techniques [8]. On the other hand, domain-specific frameworks illustrate how structured engineering of DTs can look in practice [10, 11]. However, there is still no unified view that systematically compares these different kinds of frameworks and process models, identifies their commonalities and differences, and distils from them a reference method that could be adapted across domains.

This paper presents the first stage of my research that aims to address this gap. The remainder is structured as follows: Section 2 formulates the research questions and objectives; Section 3 outlines the state of the art; Section 4 describes the research methodology; Section 5 introduces the proposed contribution; and Section 6 discusses open issues and next steps.

2 Research Questions and Objectives

The aim of this research is to develop a paradigm-independent, model-driven reference method for Digital Twin development that integrates existing frameworks and process models into a unified, domain-adaptable methodological framework.

To achieve this aim, the following research questions guide the work:

RQ1. What are the main frameworks, process models and method fragments that have been proposed for Digital Twin development, and how do they differ in scope, lifecycle coverage and abstraction level?

RQ2. What common patterns, complementarities and gaps emerge when these methods are systematically compared along well-defined dimensions?

RQ3. Can a unified reference method be synthesised that integrates recurring phases and practices, explicitly relates them to enabling technologies, and can be specialised for different domains?

These research questions are addressed through six objectives:

1. Analyse and refine the Digital Twin development problem. Clarify the scope of DT development activities, stakeholders and lifecycle phases that a development method should cover [3, 7].
2. Systematically identify representative DT development frameworks and process models. Perform a structured literature search focusing on approaches with explicit methodological guidance [6, 8, 9, 10, 11].
3. Develop a comparison framework. Define comparison dimensions (lifecycle coverage, abstraction levels, artefacts and models, MDE/ontology use, tooling, domain specificity, validation) and a uniform description schema.
4. Apply the framework to the selected methods. Produce a structured comparative analysis that surfaces common patterns, complementarities and gaps.
5. Synthesise a unified reference method. Integrate recurring phases and practices, explicitly relate them to enabling technologies (modelling languages, MDE automation, ontology tooling), and allow for domain specialisation.
6. Illustrate and preliminarily evaluate the method on a case study. Apply it to a simplified DT scenario (e.g. infrastructure or industrial systems) and reflect on its applicability, strengths and limitations.

3 State of the Art

This section reviews the main strands of literature that inform the present research: general Digital Twin surveys, methodology-focused reviews, process model comparisons and specialised model-driven or ontology-based engineering approaches. Applying the search and screening protocol detailed in Section 4 (Step 1) to IEEE Xplore, ACM DL, Scopus and SpringerLink, an initial pool of 320+ candidates was reduced to 28 primary studies. Table 1 positions the reviewed contributions along two dimensions (focus and domain scope); the subsections below summarise the key findings.

Following Carrión and Pastor [6], we distinguish two complementary kinds of methodological contributions. A framework is a structured, high-level arrangement of concepts, components and relationships that orients practitioners through the overall engineering approach without prescribing a specific activity sequence. A process model, by contrast, prescribes an ordered (possibly iterative) sequence of activities, roles and artefacts that practitioners execute to build and operate a Digital Twin. Frameworks therefore tend to emphasise structure, whereas process models emphasise execution; many real-world proposals mix both aspects, which motivates the systematic comparison pursued here.

The concept of the Digital Twin was first articulated by Grieves as a triad of a physical entity, its virtual counterpart and a bidirectional data connection [1], and refined by Tao et al. [2] into a five-dimensional model adding connection, data and services. Jones et al. [12] identify thirteen core characteristics - including real-time synchronisation, fidelity and lifecycle integration - yet report no consensus definition. Fuller et al. [4] survey enabling technologies and open challenges, noting that the heterogeneity of DT implementations is both a source of innovation and a barrier to systematic engineering.

From a development perspective, DTs are inherently interdisciplinary artefacts whose construction demands coordination across requirements, domain modelling, architecture, sensor integration, simulation, deployment and runtime operations [7, 3]. This breadth calls for systematic development methods, yet Carrión and Pastor [6] observe that explicit, reusable and well-documented methodologies remain scarce and heterogeneous.

Several recent reviews map the methodological landscape. Carrión and Pastor [6] identify two main families - high-level frameworks and detailed step-by-step process models - and conclude that generalisable methodologies remain the exception.

Sommer [9] compares fifteen DT process models, evaluating them by their phases, sequencing and degree of iteration. The study finds that largely sequential models still dominate, but hybrid and more agile approaches are gaining importance. However, the comparison remains at the process level and does not systematically assess supporting technologies or artefact types.

Fett et al. [3] broaden the scope by examining DTs alongside Cyber-Physical Systems and Product-Service Systems, categorising approaches into holistic lifecycle perspectives, architectural patterns and model-related methods. Their findings underline that approaches are distributed across communities and differ in granularity, terminology and focus—making it difficult for practitioners to select suitable methods.

A complementary perspective comes from Lehner et al. [8], who conduct a systematic mapping study on Model-Driven Engineering for Digital Twins. Their mapping confirms that MDE adoption is growing—particularly through model transformations, code generation and domain-specific languages—but that these techniques are applied in diverse ways across domains. Dalibor et al. [5] provide a cross-domain mapping of software engineering practices for DTs, identifying recurring engineering concerns (modelling, integration, deployment, evolution) that transcend individual application domains.

Beyond surveys, several contributions propose concrete engineering frameworks. Chartrain et al. [10] combine ontologies with MDE standards to enable model-driven DTs for railway infrastructure, pointing toward a future generic framework. Oakes et al. [11] propose an ontological, service-driven approach in which DTs are engineered as constellations of services, enablers, models and data, supported by composition tooling.

At a more technical level, Kirchhof et al. [13] integrate CPS architecture descriptions with DT information systems using MontiArc and UML/P. Schroeder et al. [14] propose a methodology for DT modelling and deployment aligned with Industry 4.0, emphasising layered modelling. Bibow et al. [15] develop a model-driven approach for injection-moulding DTs with a custom DSL and OPC-UA bindings. Each demonstrates methodical, tool-supported engineering, but targets a specific domain or use case.

Existing work either surveys the DT landscape at a high level or proposes specific engineering solutions. What is missing is a systematic comparison of the concrete frameworks and process models against well-defined

dimensions, followed by a synthesis into a unified reference method. This is the gap the present research aims to fill.

Table 1. Positioning of the reviewed contributions.

Focus	Domain-specific	Cross-domain / Generic
Concept surveys	-	Grieves [1], Jones et al. [12], Fuller et al. [4]
Methodology reviews	-	Carrión & Pastor [6], Sommer [9], Fett et al. [3]
MDE / SE mappings	-	Lehner et al. [8], Dalibor et al. [5]
Engineering frameworks	Chartrain et al. [10], Bibow et al. [15], Schroeder et al. [14]	Oakes et al. [11], Kirchhof et al. [13]

Analytical synthesis. Three cross-cutting observations follow from the reviewed literature. First, coverage is uneven: surveys and mappings dominate Table 1, whereas concrete cross-domain engineering frameworks remain the exception. Second, methodological guidance and enabling technologies are usually treated in separate strands, and the traceability between method phases and their supporting MDE techniques is left implicit. Third, existing comparisons operate at heterogeneous granularities - phases, MDE techniques or architectural layers - which makes them hard to combine. Together these observations motivate a middle layer that compares methods along uniform dimensions and distils the result into a reference method coupling phases with enabling technologies.

4 Research Methodology

The research combines systematic literature review, comparative method analysis and inductive synthesis, aligned with the six objectives stated in Section 2. Figure 1 illustrates the five iterative steps of the research process.

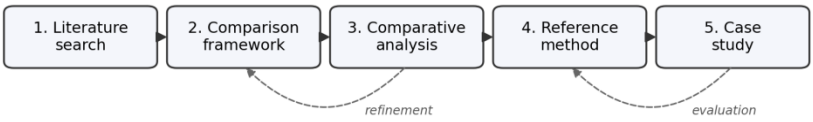

Figure 1. Research process: five iterative steps.

1. Systematic literature search and selection (Objectives 1-2): search strings combining ‚digital twin', ‚development method', ‚process model', ‚framework' and ‚model-driven engineering' were applied to IEEE Xplore, ACM DL, Scopus and SpringerLink. Inclusion criteria required explicit methodological guidance for DT development; short abstracts, position papers and duplicates were excluded. Two-stage screening (title/abstract, then full text) reduced 320+ candidates to 28 primary studies [3, 6, 8].
2. Comparison framework development (Objective 3): a set of comparison dimensions is developed iteratively, starting from criteria identified in existing reviews (lifecycle coverage, abstraction levels, artefact types, MDE and ontology usage, tooling support, domain specificity, validation approach) and refined through pilot coding of a subset of methods.
3. Structured comparative analysis (Objective 4): the comparison framework is applied to the selected methods, producing a structured analysis that highlights common patterns, complementarities and gaps. Osterweil's comparative analysis method [16] informs the systematic, dimension-by-dimension comparison.
4. Inductive synthesis of the reference method (Objective 5): recurring phases, artefacts and enabling technologies identified in the comparative analysis are inductively integrated into a single reference method. The method is expressed as a process model with accompanying guidelines, explicitly relating phases to enabling technologies.
5. Case study illustration and evaluation (Objective 6): the reference method is applied to a simplified DT scenario (e.g. infrastructure or industrial systems). Applicability, strengths and limitations are reflected upon.

Current progress. This paper reports on the first stage of my research. Objectives 1-2 are substantially complete (literature surveyed, representative methods identified - Table 1); Objectives 3-4 are underway (preliminary excerpt in Table 2); Objectives 5-6 remain for subsequent stages.

5 Proposed Contribution

The expected contributions of this research are:

1. A structured catalogue and taxonomy of existing DT development methods, classified by scope, lifecycle coverage, abstraction level and supporting technologies.

2. A reusable comparison framework with well-specified dimensions and a uniform description schema, usable by other researchers and practitioners to position new approaches.
3. A unified reference method for DT development, expressed as a process model with accompanying guidelines. It integrates recurring phases and practices, explicitly relates them to enabling technologies (modelling languages, MDE automation, ontology tooling, DT platforms), and allows for domain specialisation.
4. A small-scale case study that demonstrates how the reference method is instantiated in practice and provides initial feedback on usability.

The comparison framework under development organises the analysis along the following preliminary dimensions, derived from the reviewed literature:

- Structural dimensions: lifecycle coverage (phases addressed - requirements, design, implementation, deployment, operation, evolution); abstraction level (framework vs. process model, relation to the MDA four-layer architecture [17]); artefacts and modelling languages produced; degree of MDE automation (transformations, code generation, model interpretation [8, 18]).
- Supporting dimensions: ontology and semantic support [10, 11]; tooling and DT platform dependencies; domain specificity and adaptability; validation approach (case study, experiment, industrial pilot, expert review).

Preliminary findings. Although the comparison is ongoing, the initial excerpt already yields partial answers. RQ1 is partly addressed by Tables 1-2: frameworks and process models differ markedly in scope (domain-specific vs. cross-domain), lifecycle coverage (design-heavy vs. full-lifecycle) and in the MDE support they assume (code generation, DSLs or ontology-enhanced transformations). At the pattern level, RQ2 is addressed by the observation that no single reviewed method covers the whole DT lifecycle and that the coupling between phases and enabling technologies is typically left implicit. RQ3 will be answered once the comparison is completed and the inductive synthesis (Section 4, Step 4) is performed.

Table 2. Preliminary comparison excerpt (four representative methods).

Method	Lifecycle	MDE use	Domain	Validation
Kirchhof et al. [13]	Design, impl.	Code gen.	CPS (generic)	Case study
Bibow et al. [15]	Design, depl.	DSL, code gen.	Manufacturing	Industrial pilot
Schroeder et al. [14]	Full lifecycle	Layered MDE	Industry 4.0	Proof of concept
Chartrain et al. [10]	Design, impl.	Ontology+MDE	Railway	Case study

6 Open Issues and Next Steps

Several open issues remain to be addressed:

- Comparison framework completeness. The dimension set is derived from existing reviews and may need refinement as additional methods are analysed; a principled stopping criterion is still to be defined.
- Synthesis approach. Integrating diverse methods into a single reference method requires design decisions about granularity, prescriptiveness and domain adaptability - balancing generality with actionable specificity.
- Case study selection. Choosing a scenario that exercises the full DT lifecycle while remaining tractable will strongly shape which aspects of the reference method can be evaluated.

Acknowledgments

The author is grateful to the research supervisor Dr. Audronė Lupeikienė (Vilnius University) for guidance and support throughout this research.

References

[1] Grieves, M., Vickers, J. (2017). Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In: Transdisciplinary Perspectives on Complex Systems, Springer, pp. 85–113.

[2] Tao, F., Zhang, H., Liu, A., Nee, A. Y. C. (2019). Digital Twin in Industry: State-of-the-Art. IEEE Transactions on Industrial Informatics, 15(4), 2405–2415.

[3] Fett, M., Wilking, F., Goetz, S., Kirchner, E., Wartzack, S. (2023). A literature review on the development and creation of digital twins, cyber-physical systems, and product-service systems. Sensors, 23(24), 9786.

[4] Fuller, A., Fan, Z., Day, C., Barlow, C. (2020). Digital Twin: Enabling Technologies, Challenges and Open Research. IEEE Access, 8, 108952–108971.

[5] Dalibor, M., Michael, J., Rumpe, B., Varga, S., Wortmann, A. (2022). A cross-domain systematic mapping study on software engineering for Digital Twins. Journal of Systems and Software, 193, 111361.

[6] Carrión, E., Pastor, Ó. (2023). A systematic review of methodologies for developing digital twins: Insights and recommendations for effective implementation. In: Practice of Enterprise Modeling – PoEM 2023 Companion Proceedings.
[7] Metzger, M., Wedel, F.-H. (2023). A Systematic Literature Review on Digital Twins. Seminar Paper, HEC Paris / University of Bayreuth.
[8] Lehner, D., Zhang, J., Pfeiffer, J., Sint, S., Splettsstößer, A.-K., Wimmer, M., Wortmann, A. (2025). Model-driven engineering for digital twins: a systematic mapping study. Software and Systems Modeling, 1–39.
[9] Sommer, L. (2024). Digital twin modeling: A comparison of current approaches. Open Research Europe, 4(56), 56.
[10] Chartrain, A., Dessagne, G., Haddad, N., Hill, D. R. C. (2024). Linking Digital Twin Design and Ontologies with Model-Driven Engineering. In: Proc. IC3K 2024, vol. 2, pp. 265–276.
[11] Oakes, B., Gomes, C., Kamburjan, E., Abbiati, G., Ecem Bas, E., Engelsgaard, S. (2024). Towards Ontological Service-Driven Engineering of Digital Twins. In: Proc. MODELS 2024, ACM, pp. 464–469.
[12] Jones, D., Snider, C., Nassehi, A., Yon, J., Hicks, B. (2020). Characterising the Digital Twin: A systematic literature review. CIRP Journal of Manufacturing Science and Technology, 29, 36–52.
[13] Kirchhof, J. C., Michael, J., Rumpe, B., Varga, S., Wortmann, A. (2020). Model-driven Digital Twin Construction. In: Proc. MODELS 2020, pp. 90–101.
[14] Schroeder, G. N., et al. (2021). A Methodology for Digital Twin Modeling and Deployment for Industry 4.0. Proceedings of the IEEE, 109(4), 556–567.
[15] Bibow, P., et al. (2020). Model-Driven Development of a Digital Twin for Injection Molding. In: CAiSE 2020, LNCS 12127, Springer, pp. 85–100.
[16] Song, X., Osterweil, L. J. (1992). Toward objective, systematic design-method comparisons. IEEE Software, 9(3), 43–53.
[17] Object Management Group. (2014). MDA Guide rev. 2.0. Technical Report ormsc/2014-06-01.
[18] Brambilla, M., Cabot, J., Wimmer, M. (2017). Model-Driven Software Engineering in Practice. 2nd ed. Morgan & Claypool.