Toward a Paradigm-Independent Model-Driven Method for Digital Twin Development

Anton Zagzin[1]

[1]Vilnius University, Faculty of Mathematics and Informatics, Naugarduko g. 24, LT-03225 Vilnius, Lithuania

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, focus on particular dimensions such as methodologies or Model-Driven Engineering techniques, or propose domain-specific engineering approaches. However, no unified view systematically compares these different kinds of contributions, identifies their commonalities and differences, and distils from them a paradigm-independent, model-driven reference method that could be adapted across domains. This paper presents my research that aims to move toward such a unified method for Digital Twin development by performing a systematic comparison of existing frameworks and process models, and by synthesising their results into a coherent reference method. The research methodology combines systematic literature review, comparative method analysis, and inductive synthesis. This paper reports on the current progress, outlines the proposed contribution, and discusses open issues and next steps.

Keywords

Digital Twin, development method, Model-Driven Architecture, paradigm-independent, 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 service-oriented business models [3, 4].

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 [5]. 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, there is a growing recognition that Digital Twins cannot be engineered in an ad-hoc manner. Instead, systematic development methods, frameworks and process models are needed to guide practitioners through the main activities and decisions involved in creating and evolving DTs. Several research articles explicitly focus on this methodological dimension. A systematic review by Carrión and Pastor analyses methodologies for developing Digital Twins and identifies two main families of contributions: structured high-level frameworks and more detailed step-by-step process models [6]. Other reviews adopt broader perspectives: Metzger provides a systematic literature review summarising core concepts, application areas, implementation strategies and challenges [7]; Fett et al. examine DTs together with Cyber-Physical Systems and Product-Service Systems, categorising development approaches into holistic lifecycle perspectives, architectural patterns and model-related

Baltic DB&IS 2026 Conference Forum and Doctoral Consortium, 28 June - 1 July 2026, Tartu, Estonia

anton.zagzin@mif.stud.vu.lt (A. Zagzin)

methods [3]. At the same time, more specialised works address particular aspects. Lehner et al. map the use of Model-Driven Engineering (MDE) automation techniques—model transformations, code generation and model interpretation—in DT contexts [8]. Sommer compares fifteen process models regarding their phases, sequencing and degree of iteration [9]. Complementary research proposes 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. Specifically, the limitations of existing surveys can be characterised in three aspects: Carrión and Pastor [6] catalogue methodologies but do not relate development phases to the enabling technologies that realise them; Sommer [9] compares process models on phase sequencing but treats MDE automation, ontologies and tooling as out of scope; Lehner et al. [8] systematically map MDE techniques in DT contexts but stop short of embedding them in a lifecycle method. Practitioners building an information-system-level DT—e.g. for an urban-infrastructure or industrial-asset platform—therefore have no single source that prescribes which phases, artefacts and MDE techniques to combine.

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?

Each research question is mapped to a specific gap identified in Section 1 and to a distinct expected contribution in Section 5: RQ1 addresses the catalogue/positioning gap and is answered by Contribution C1 (structured catalogue and taxonomy, anchored in Table 1); RQ2 addresses the cross-method comparison gap and is answered by Contribution C2 (reusable comparison framework); RQ3 addresses the synthesis gap and is answered by Contribution C3 (the paradigm-independent reference method), with Contribution C4 (the case study) providing the first illustrative validation.

The research questions are addressed through six objectives, grouped into three blocks that correspond to distinct research outcomes rather than a single linear workflow—*Foundation* (O1–O2, supporting C1–C2), *Construction* (O3–O5, supporting C2–C3) and *Validation* (O6, supporting C4):

1. **Analyse and refine the Digital Twin development problem.** Based on existing conceptions of Digital Twins, clarify the scope of DT development activities, stakeholders and lifecycle phases that a development method should cover [3, 12].

2. **Systematically identify and select representative DT development frameworks and process models.** Building on existing reviews and mapping studies, perform a structured literature search and selection, focusing on approaches that provide explicit methodological guidance [6, 10, 8, 11, 9].
3. **Develop a comparison framework for DT development methods.** Define a set of comparison dimensions and criteria (such as lifecycle coverage, abstraction levels, artefacts and models, MDE and ontology usage, tooling support, domain specificity and validation) and a corresponding schema for describing methods in a uniform way.
4. **Apply the comparison framework to the selected methods.** Produce a structured comparative analysis that highlights common patterns, complementarities and gaps across existing frameworks and process models.
5. **Synthesise a unified reference method for DT development.** On the basis of the comparison, propose a method that integrates recurring phases and practices, explicitly relates them to enabling technologies (e.g. modelling languages, MDE automation, ontology tooling) and can be specialised for different domains.
6. **Illustrate and preliminarily evaluate the reference method on a case study.** Apply the method to a simplified DT scenario, for example in the domain of infrastructure or industrial systems, and reflect on its applicability, strengths and limitations.

3. State of the art

The main concepts of this research are as follows. Following Carrión and Pastor [6], this paper distinguishes four terms that can be interpreted differently. 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* prescribes an ordered (possibly iterative) sequence of activities, roles and artefacts that practitioners execute to build and operate a Digital Twin. A *methodology* is the higher-order discipline of selecting, justifying and applying frameworks and process models. A *reference method*—the artefact targeted by this research—is a prescriptive process model with accompanying guidelines and a meta-model of phase $\leftrightarrow$ enabling-technology relationships, designed to be instantiated and specialised for different domains.

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 and ontology-based engineering approaches. Papers were identified through systematic search in IEEE Xplore, ACM DL, Scopus and SpringerLink using combinations of the terms *digital twin*, *development method*, *process model*, *framework*, and *model-driven engineering*. The initial search yielded over 320 results; after screening by title and abstract and applying inclusion criteria (explicit methodological guidance for DT development, English language, peer-reviewed), 28 primary studies were selected for detailed analysis, of which 18 were retained for deep dimension-by-dimension comparison (see Section 4).

3.1. Digital twin concepts and development challenges

The concept of the Digital Twin was first articulated by Grieves in the context of product lifecycle management as a triad of a physical entity, its virtual counterpart, and a bidirectional data connection [1]. Tao et al. [2] refined this into a five-dimensional model comprising physical entity, virtual model, connection, data and services. Jones et al. [12], in a systematic literature review, identified thirteen core characteristics including real-time synchronisation, fidelity of representation and lifecycle integration, while noting that no single definition has achieved consensus across communities. Fuller et al. [4] survey enabling technologies and open challenges, highlighting that the heterogeneity of DT implementations is both a source of innovation and a barrier to systematic engineering.

From a development perspective, the key challenge is that DTs are inherently interdisciplinary artefacts. Building a DT requires coordinating requirements engineering, domain modelling, system

architecture, sensor integration, simulation, deployment and runtime operations [5, 3]. This breadth of concerns calls for systematic development methods, yet as Carrión and Pastor [6] observe, explicit, reusable and well-documented methodologies remain relatively scarce and heterogeneous.

3.2. Methodology and process model reviews

Several recent reviews have begun to map the methodological landscape. Carrión and Pastor [6] identify two main families: structured high-level frameworks that define the overall approach, and more detailed step-by-step process models that prescribe sequences of activities. Their review concludes that while many authors describe how a particular DT was built in a given context, generalisable methodologies remain the exception rather than the rule.

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.

3.3. Specialised frameworks and approaches

Beyond surveys, several contributions propose concrete engineering frameworks. Chartrain et al. [10] present a framework that combines ontologies with MDE standards to enable model-driven DTs for railway infrastructure, arguing that such an approach can contribute to 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, with workflows supported by tooling guiding practitioners in composing and deploying DT services.

At a more technical level, Kirchhof et al. [13] integrate CPS architecture descriptions with DT information systems using MontiArc and UML/P, demonstrating how MDE can bridge systems engineering and software engineering perspectives. Schroeder et al. [14] propose a methodology for DT modelling and deployment aligned with Industry 4.0, emphasising layered modelling and MDE conventions. Bibow et al. [15] develop a model-driven approach for injection-moulding DTs using a custom DSL and OPC-UA service bindings. Each of these works demonstrates the potential of methodical, tool-supported engineering, but they are typically tailored to specific domains or use cases.

3.4. Identified gap

Table 1 positions the reviewed contributions along two dimensions: their scope (from specific domain to generic/cross-domain) and their focus (from DT concept surveys through methodology reviews to concrete engineering frameworks).

The table reveals that existing work either surveys the landscape at a high level (top rows) or proposes specific engineering solutions (bottom row). What is missing is the *middle layer*: 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. By bridging descriptive surveys and prescriptive frameworks, the present work intends to produce a

Table 1
Positioning of 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]

reusable comparison framework and a reference method that integrates recurring phases, artefacts and enabling technologies in a domain-adaptable way.

4. Research methodology

The research follows five iterative steps, aligned with the six objectives stated in Section 2. Figure 1 illustrates the overall research process.

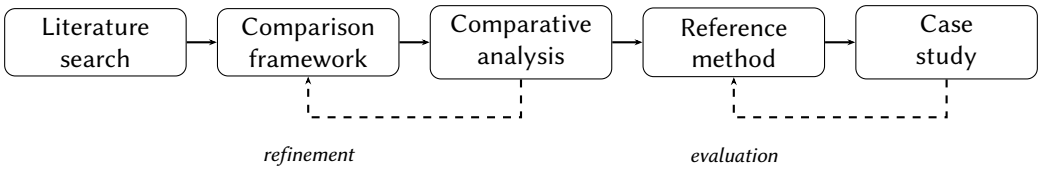

Figure 1: Research process: five iterative steps mapping to the six research objectives.

1. **Systematic literature search and selection** (Objectives 1–2): search strings and inclusion criteria are defined following established systematic review guidelines [6, 3, 8]. Academic databases (IEEE Xplore, ACM DL, Scopus, SpringerLink) are searched, and studies are screened and selected based on whether they provide explicit methodological guidance for DT development. The explicit review protocol is: (i) *inclusion criteria*—peer-reviewed, English-language publications that prescribe at least one phase of DT development with associated artefacts or enabling technologies; (ii) *exclusion criteria*—purely conceptual position papers, short abstracts, duplicates, and publications that describe a single DT instance without methodological abstraction; (iii) *coding scheme*—each retained study is labelled by {focus, lifecycle scope, MDE use, ontology use, domain, validation}; (iv) *stopping criterion*—the comparison dimension set is frozen when three consecutive newly coded studies introduce no new codes; (v) *audit trail*—all selection decisions are logged in a spreadsheet recoverable from the dissertation repository.
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): the unified reference method is iteratively constructed based on the patterns, recurring phases and enabling technologies identi-

fied in the comparative analysis. The method is expressed as a process model with accompanying guidelines, explicitly relating phases to enabling technologies. The synthesis is governed by three explicit rules: (a) *pattern coding*—recurring activities and artefacts are clustered across the coded studies, with each cluster requiring support from at least two independent source methods before being admitted into the reference method; (b) *conflict resolution*—when sources prescribe contradictory orderings, the one with the stronger phase ↔ enabling-technology coupling (i.e. the one that makes the dependency explicit) is retained; (c) *traceability/audit trail*—every phase, activity, artefact and enabling technology in the resulting reference method is traceable to at least one source study via the coding spreadsheet, so that readers can reproduce the synthesis path.

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. The evaluation roadmap is three-tiered: (1) instantiation on a single illustrative case study (this paper's Objective 6, planned for the subsequent work); (2) expert walk-through with three to five DT and MDE researchers using a structured feedback instrument; (3) optional follow-up multi-case comparison and limited Delphi-style validation, if time and access permit. The single-case starting point is deliberate at this point in the research; tiers (2) and (3) are explicit limitations rather than completed work.

Current progress. This paper presents my research as a whole; the present focus is on the early objectives, while the synthesis and case-study evaluation are carried out in subsequent work. Concretely, the SLR funnel has narrowed 320+ candidates to 28 primary studies, of which 18 have so far been coded for deep dimension-by-dimension comparison; 8 comparison dimensions are currently active in the framework (lifecycle coverage, abstraction levels, artefacts and models, MDE automation, ontology and semantic support, tooling and platform support, domain specificity, validation); and three cross-cutting patterns have emerged so far—uneven lifecycle coverage, separation of methodological guidance from enabling technologies, and heterogeneous comparison granularities—each of which will be reported in detail in the comparative-analysis paper currently in preparation. Objectives 1 and 2 are substantially complete: the literature has been surveyed, representative frameworks and process models have been identified, and the comparison dimensions have been drafted. Work on Objectives 3 and 4—developing and applying the comparison framework—is underway. Objectives 5 and 6 (synthesis and case analysis) are the objects of future work.

5. Proposed contribution

The contribution targeted by this research is a *prescriptive process model with accompanying guidelines and a meta-model of phase ↔ enabling-technology relationships*. It is not a taxonomy, not a survey, and not an information-system meta-model alone. The four sub-contributions below (C1–C4) are therefore distinct outcomes, each with its own evaluation approach.

The expected contributions of this research are:

1. (**C1**) A **structured catalogue and taxonomy** of existing Digital Twin development methods, classified by scope, lifecycle coverage, abstraction level and supporting technologies. *Evaluation:* completeness check against the methodologies covered by existing reviews (Carrión & Pastor, Sommer, Fett et al., Lehner et al.), confirming that no method present in those reviews is missing from the catalogue and that the dimension set discriminates them meaningfully.
2. (**C2**) A **reusable comparison framework** that can be used by other researchers and practitioners to analyse and position new approaches. The framework defines well-specified dimensions and a uniform description schema for DT development methods. *Evaluation:* a coding-reliability pilot in which two coders independently classify a 4–6-method sub-sample and the inter-coder agreement is measured, plus a dimension-stability test against the stopping criterion of Section 4 (Step 1).

3. (**C3**) A **paradigm-independent, model-driven reference method** for Digital Twin development, expressed as a process model grounded in MDA principles [17] and accompanied by methodological guidelines. The reference method integrates recurring phases and practices identified across the analysed approaches, explicitly relates them to enabling technologies (modelling languages, MDE automation, ontology tooling, DT platforms), and can be specialised for different domains without being tied to a particular modelling paradigm. *Evaluation:* an expert walk-through with three to five DT and MDE researchers, complemented by instantiation on the case study below; the walk-through assesses clarity, completeness and prescriptive strength of each phase $\leftrightarrow$ enabling-technology mapping.
4. (**C4**) A **small-scale case study** that demonstrates how the reference method can be instantiated in practice and provides initial feedback on its usability. *Evaluation:* structured reflection along applicability, prescriptive value and adaptability, following established case-study research guidelines in software engineering; limitations of single-case evidence are explicitly acknowledged.

5.1. Preliminary comparison dimensions

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

- **Lifecycle coverage**: which phases of DT development does the method address (requirements, design, implementation, deployment, operation, evolution, decommissioning)?
- **Abstraction levels**: does the method operate at a high-level framework level, a detailed process model level, or both? How does it relate to the MDA four-layer architecture [17]?
- **Artefacts and models**: what types of artefacts are produced (domain models, metamodels, platform models, simulation models, data schemas)? What modelling languages are used?
- **MDE automation**: does the method employ model transformations, code generation, model interpretation, or other MDE techniques [8, 18]?
- **Ontology and semantic support**: does the method leverage ontologies for formalising domain knowledge, enabling reasoning, or supporting interoperability [10, 11]?
- **Tooling and platform support**: what tools and DT platforms does the method assume or integrate with?
- **Domain specificity**: is the method generic or tailored to a specific domain? How adaptable is it?
- **Validation**: how has the method been validated (case study, experiment, industrial pilot, expert review)?

The preliminary application of these dimensions to a subset of reviewed methods has already revealed that no single existing method covers all lifecycle phases; most methods are either strong on early phases (requirements, design) or on later phases (deployment, operation), but rarely both. Similarly, the relationship between methodological guidance and enabling technologies is typically implicit, which limits traceability and reuse.

Table 2 illustrates a preliminary excerpt of the comparison applied to four representative approaches.

To make the artefact concrete, an instance of the reference method is composed, for each lifecycle phase, of four elements: (a) *phase* (e.g. *Design*), (b) *prescribed activities* (e.g. *specify physical entity*, *define virtual model*, *specify connection schema*), (c) *produced artefacts* (e.g. domain meta-model, platform model, data schema), and (d) *enabling technologies* (e.g. a domain-specific modelling language, an MDA platform-independent-to-platform-specific transformation, an ontology binding). A follow-up paper will provide the full reference method along with worked examples for at least two lifecycle phases on the case-study domain.

6. Open issues and next steps

Several open issues remain to be addressed:

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

- **Comparison framework completeness.** The current set of comparison dimensions is derived from existing reviews, but may need refinement as it is applied to additional methods. A principled stopping criterion for the dimension set needs to be defined.
- **Synthesis approach.** Integrating findings from diverse methods into a single unified reference method involves design decisions about granularity, prescriptiveness and domain adaptability. Balancing generality (applicable across domains) with specificity (actionable for practitioners) is a key challenge.
- **Case study selection.** The choice of domain and scenario for the illustrative case study will influence which aspects of the reference method are tested. Selecting a case that exercises the full lifecycle while remaining tractable is non-trivial.
- **Validation scope.** Beyond the single case study, broader validation with domain experts or practitioners would strengthen the results but may exceed the current research scope.

Limitations. The study is limited to English-language publications indexed in four major databases, which may exclude relevant grey literature or publications in other languages. The comparison dimensions, while grounded in existing reviews, reflect the author's analytical perspective and may evolve as additional methods are analysed. A single case study will provide illustrative rather than generalisable evidence of the reference method's applicability; the three-tier evaluation roadmap introduced in Section 4 (Step 5) is the planned mitigation, although tiers (2) and (3) remain forward-looking rather than completed work.

Acknowledgments

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

Declaration on Generative AI

During the preparation of this work, the author used Claude (Anthropic, Opus 4.7) and GitHub Copilot in order to: grammar and style review, and paraphrasing of author-drafted text. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] M. Grieves, J. Vickers, Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems, in: Transdisciplinary Perspectives on Complex Systems, Springer, 2017, pp. 85–113. doi:10.1007/978-3-319-38756-7_4.

[2] F. Tao, H. Zhang, A. Liu, A. Y. C. Nee, Digital twin in industry: State-of-the-art, IEEE Transactions on Industrial Informatics 15 (2019) 2405–2415. doi:10.1109/TII.2018.2873186.

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

[4] A. Fuller, Z. Fan, C. Day, C. Barlow, Digital twin: Enabling technologies, challenges and open research, IEEE Access 8 (2020) 108952–108971. doi:10.1109/ACCESS.2020.2998358.

[5] M. Dalibor, J. Michael, B. Rumpe, S. Varga, A. Wortmann, A cross-domain systematic mapping study on software engineering for digital twins, Journal of Systems and Software 193 (2022) 111361. doi:10.1016/j.jss.2022.111361.

[6] E. Carrión, Ó. Pastor, A systematic review of methodologies for developing digital twins: Insights and recommendations for effective implementation, in: Practice of Enterprise Modeling – PoEM 2023 Companion Proceedings, 2023.

[7] M. Metzger, F.-H. Wedel, A Systematic Literature Review on Digital Twins, Seminar Paper, HEC Paris / University of Bayreuth, 2023. Seminar IT-Management in the Digital Age.

[8] D. Lehner, J. Zhang, J. Pfeiffer, S. Sint, A.-K. Splettstößer, M. Wimmer, A. Wortmann, Model-driven engineering for digital twins: a systematic mapping study, Software and Systems Modeling (2025) 1–39. doi:10.1007/s10270-024-01230-5.

[9] L. Sommer, Digital twin modeling: A comparison of current approaches, Open Research Europe 4 (2024) 56.

[10] A. Chartrain, G. Dessagne, N. Haddad, D. R. Hill, Linking digital twin design and ontologies with model-driven engineering: Application to railway infrastructure, in: Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2024), volume 2, 2024, pp. 265–276.

[11] B. Oakes, C. Gomes, E. Kamburjan, G. Abbiati, E. Ecem Bas, S. Engelsgaard, 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 2024), ACM, 2024, pp. 464–469.

[12] D. Jones, C. Snider, A. Nassehi, J. Yon, B. Hicks, Characterising the digital twin: A systematic literature review, CIRP Journal of Manufacturing Science and Technology 29 (2020) 36–52. doi:10.1016/j.cirpj.2020.02.002.

[13] J. C. Kirchhof, J. Michael, B. Rumpe, S. Varga, A. Wortmann, Model-driven digital twin construction: Synthesizing the integration of cyber-physical systems with their information systems, in: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS 2020), 2020, pp. 90–101. doi:10.1145/3365438.3410941.

[14] G. N. Schroeder, C. Steinmetz, R. N. Rodrigues, C. E. Pereira, D. B. Espindola, R. Henriques, A. Pretto, M. Ventura, I. Costa, A methodology for digital twin modeling and deployment for industry 4.0, Proceedings of the IEEE 109 (2021) 556–567. doi:10.1109/JPROC.2020.3032444.

[15] P. Bibow, M. Dalibor, C. Hopmann, B. Mainz, B. Rumpe, D. Schmalzing, M. Schmitz, A. Wortmann, Model-driven development of a digital twin for injection molding, in: Advanced Information Systems Engineering – CAiSE 2020, volume 12127 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 85–100. doi:10.1007/978-3-030-49435-3_6.

[16] X. Song, L. Osterweil, Toward objective, systematic design-method comparisons, IEEE Software 9 (1992) 43–53. doi:10.1109/52.136166.

[17] Object Management Group, MDA Guide rev. 2.0, Technical Report ormsc/2014-06-01, Object Management Group, 2014. URL: https://www.omg.org/cgi-bin/doc?ormsc/14-06-01.

[18] M. Brambilla, J. Cabot, M. Wimmer, Model-Driven Software Engineering in Practice, 2nd ed., Morgan & Claypool Publishers, 2017. doi:10.2200/S00751ED2V01Y201701SWE004.