Formal Methods meet Digital Twins: Challenges and Opportunities

Einar Broch Johnsen[1], Eduard Kamburjan[2,1], Andrea Pferscher[1], and Silvia Lizeth Tapia Tarifa[1]

[1] University of Oslo, Oslo, Norway
{einarj,andreapf,sltarifa}@uio.no
[2] IT University of Copenhagen, Copenhagen, Denmark
eduard.kamburjan@itu.dk

Abstract. The advent of digital twins gives us an opportunity to reflect on the relationship between models and modelled systems. We may think of digital twins not merely as models, but as systems for model management, integration, and composition. In fact, digital twins are model-centric systems that maintain a two-way connection between an ecosystem of models and the modelled system, realised through streams of observations and streams of interventions. This connection introduces agility as the digital twin can typically both adapt its models on-the-fly to changes in a modelled system and influence the modelled system's behaviour. In this paper, we discuss key concepts of digital twins from a formal methods perspective and suggest opportunities and challenges for formal methods in digital twin systems. In particular, we consider how formal techniques can be integral to the digital twin, both in terms of digital twin technology and in terms of digital twin models, as well as notions of correctness for the digital twin itself.

1 Introduction

Today, digital twins (DTs) are subject to a fair amount of hype[1] for their potential to improve efficiency and mitigate failure in a broad range of systems, during system operation. DTs are a key concept in Industry 4.0 [10,54]; applications of DTs are found across engineering disciplines, based on the idea of creating an increasingly accurate "virtual replica" of a cyber-physical system to predict behaviour by means of sophisticated simulation techniques and a closed feedback loop to the actual system (e.g., [15]). DTs are now increasingly found in application domains beyond engineering, including medicine [42], healthcare [55], energy [44], manufacturing [6,40], transportation [11], and software systems [1].

DTs are useful to explain unexpected incidents, for short-term decision-making and for long-term strategic planning. To this aim, a DT can have the ability to deliver different analytical services, including *historical* analysis (what

[1] https://www.weforum.org/stories/2024/06/digital-twins-and-industrial-clusters-are-about-to-change-the-face-of-manufacturing/

happened in the past), *descriptive* analysis (what just happened), *predictive* analysis (what do we expect to happen next), *prescriptive/proactive* analysis (strategic planning, what can we do to change the expected behaviour), and *reactive* analysis (provide an immediate response for what to do now).

DTs realise these services by enabling a target system and its models to interact at runtime; e.g., models in the DT and observations from the real system work together to drive analytical services. Conceptually, DTs represent a shift from model-based to model-centric system design, and hence from a correctness-preserving perspective on system construction to a correctness-adapting perspective on system maintenance. The models and our ability to automatically analyse them, are integral to the target system, rather than a means to develop this target system. Consequently, the lifetime of a DT matches the lifetime of the actual system: we may need the ability to automatically adapt our models when these need better alignment with observed data from the actual system, and to analyse new or adapted models on-the-fly.

Formal methods are different techniques to mathematically specify and verify system behaviour, in which systems are modelled as mathematically defined structures [5, 60, 61]; these methods are interesting for the strength of the guarantees they can provide, allowing the presence (or absence) of a particular behaviour to be mathematically proven for the given model. Thus, formal methods are complementary to testing- or simulation-based analysis techniques [20]. Formal methods can be applied at different stages of system design. In terms of industrial applications, formal methods have traditionally been used for safety-critical applications, but there is an increasing uptake in other domains [5, 16].

In this paper, we argue that DTs may be of significant interest to developers of formal methods and that DTs give us an opportunity to revisit how we think formal models may be developed and used. We consider two perspectives on how formal methods and DTs can come together: how formal methods can be used as components in DTs (Sect. 3) and how formal methods can be used to analyse DTs (Sect. 4). For each perspective, we outline some open research challenges related to DTs and formal methods in a broad sense, and hint at how we have started to approach these challenges in our own work.

2 What are Digital Twins?

DTs are virtual information constructs that capture the structure, context, and behaviour of the system they are twinning, are dynamically updated with data from the actual system, have predictive capability, and inform decisions that realise value, according to a recent definition by the National Academy for Science, Engineering and Medicine (NASEM) [46]. This notion of a DT puts less weight on bidirectionality (i.e., the reactive control of a cyber-physical system) and emphasises the tight integration between a model and the modelled system to provide services for, e.g., analysis, diagnosis, prediction, fault detection and strategic planning [43].

An essential aspect of a DT is its *life cycle*: DTs are intended to be long-lived systems. Often, the DT predates the target system as a design or construction artefact that is later transformed into an operational tool, and may persist also after the actual system has ceased to operate. It closely mimics the life cycle of the actual system; specifically, this means that it needs to adapt its models to changes in its environment, which can include both the actual system and the requirements to the actual system. In fact, the user requirements to the DT may also change over the lifetime of the DT [43]; i.e., the *purpose* of the DT is likely to evolve over time. This way, the DT becomes a self-adaptive system for advanced model management, generating and adjusting its models and determining the analyses to be performed using these models on-the-fly.

A self-adaptive system typically consists of a managed and a managing layer, often organised as a MAPE-K feedback loop [58], in which *m*onitor-, *a*nalyse-, *p*lan- and *e*xecute-components interact with a *k*nowledge base. The knowledge base is often used to provide a context for the MAPE-components, capturing, e.g., historical data, domain knowledge and requirements. While the DT can be seen as a managing layer for its target system, self-adaptation within the DT can be addressed by adding an additional layer of self-adaptation to capture how the twin itself evolves over time [29], see Fig. 1. Remark that the knowledge base may include so-called *runtime models* [9] that blur the distinction between software development and execution [4], enabling introspection in support of the self-adaptation process.

The purpose of *behavioural self-adaptation* in Layer 1 is to control how the current models of the DT capture the behaviour of the actual system (thus addressing the so-called real-to-sim or reality gap, e.g. [7, 56, 62]) and make adjustments to the actual system if needed. In the feedback loop of Layer 1, the monitor collects data from the sensors connected to the actual system, the analyser assesses whether these observations of the actual system comply with requirements, the planner determines whether changes to the actual system are needed and the executor manipulates the actuators to influence the behaviour of the actual system.

The purpose of *structural self-adaptation* in Layer 2 is to control how the DT captures the structure and context of the actual system. In the feedback loop of Layer 2, the analyser determines that the requirements to the actual system have changed, that the targeted behaviour is different and that the means to achieve this behaviour are no longer the same, leading to changes in the components of Layer 1 of the DT, including its models.

3 Formal Methods in Digital Twins

Let us first consider how formal methods can be integrated in the analysis services of the DT. One important aspect of this integration, is that we are dealing with an *open environment*: we do not assume that the models perfectly reflect the actual system. Instead, analysis inside the DT is data-driven; i.e., the model configuration at a given point in time depends on the stream of observations

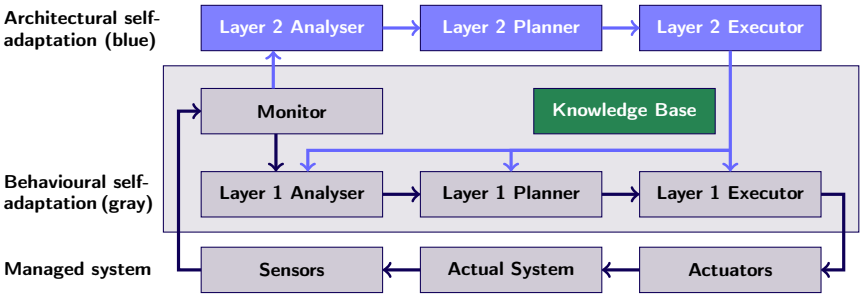

Fig. 1. A two-layered self-adaptive DT architecture with MAPE-K feedback loops for behavioural (Layer 1, gray) and architectural (Layer 2, blue) self-adaptation.

that the twin receives from the actual system. Hence, our methods need to support the dynamic configuration and composition of models reflecting the current structure and observed state of the actual system. The DT needs a strategy for how that should be done, depending on the analysis service it delivers.

Challenge 1 *How can formal models be dynamically configured and composed?*

We can understand a DT as a model management system that, driven by streams of observations of the actual system, uses contextual information to decide how to configure its models to deliver analysis services. We can build on the experiences from model management systems for *consistency* [52], and consider formal methods from this perspective as well. If we assume that the contextual information is captured in its knowledge base, we can see the DT as a model orchestrator that can compose different models. Simulation models can be orchestrated using co-simulation algorithms [17], and dynamically deployed [19]. Similar workflows may be conceived to configure and compose heterogeneous formal models, and their corresponding analyses. For example, BedreFlyt [35,51] is a DT for hospital ward planning that uses incoming patient data as its stream of observations from the ward and information about treatments stored in the knowledge base, to dynamically configure and deploy an actor-model in ABS [28], which outputs a stream of resource requirements reflecting day-to-day bed bay needs for the patients in the ward. These are, together with constraints on bed bay capacity in the ward derived from the knowledge base, used to dynamically configure a stream of optimisation problems, which are given to Z3 [8]. The twin's output is a stream of bed bay allocations that can be given to the ward's admitting nurse. This way, the twin combines several formal methods.

Challenge 2 *How can formal methods help in automatically adapting digital twin models to life cycle changes in the target system?*

We all know how challenging it can be to manually create adequate formal models [60]. Already for standardised communication protocols, ambiguities allow for different behaviours, which may require different model representations;

this suggests that incremental approaches are needed in an open world setting in which not all behaviours are known in advance [41]. Manually modelling all these different representations would be tedious and error-prone. Here, one interesting direction is to explore model learning [53], a technique to automatically generate finite state models from system traces, to automatically generate and test models in the model ecosystem of DTs [47,57]. However, for bilateral synchronisation between the DTs and real system, the underlying models must reflect the behaviour of the actual system. One way to address this problem can be to use techniques from probabilistic model checking [3] to support the model management process of the DT in selecting models based on the current environment [38].

Challenge 3 *How can formal methods enhance a digital twin by providing advanced behavioural insights into its target system?*

Formal methods can provide worst-case analyses as well as statistical guarantees not readily available with symbolic or mechanistic models. Given that the actual system is not fully understood, or that it operates in an uncontrolled environment, it seems interesting to not only analyse stream of observations from the actual system using expressive runtime verification techniques [39], but also to investigate the generation of such analysis problems on-the-fly, driven by the these stream of observations. As a step in this direction, we have explored the use of model checking techniques over sliding window segments of event streams to analyse how properties of learnt models evolve over time [36,37]. Another interesting research direction is to use formal specifications to define input scenarios amenable to a hypothetical (what-if) analysis; e.g., in the BedreFlyt DT, we have used this kind of technique to analyse average case and worst-case resource usage for bed bay allocation in a hospital ward [35].

4 Formal Methods for Digital Twins

Let us next consider formal methods for the overall DT architecture and focus on the DT's need to be dynamically updated, driven by data from the actual system, i.e., its self-adaptive capabilities. Early work on formal methods to model and analyse self-adaptive systems was surveyed by Weyns et al. [59], who reports a "remarkably low" number of papers on this topic. By structuring the configuration space of the self-adaptive system, self-adaptation has connections to variability and especially to dynamic software product lines [18, 21] in the sense that they provide a means to reconfigure the system between different configurations. While software product lines have been formalised using, e.g., featured transition systems [13], formal methods for dynamic software product lines have received less attention [49]. However, tools such as ProFeat [12], which supports family-based model-checking with a feature controller that can activate and deactivate features in the configuration space, have been used to formally model and analyse self-adaptive systems (e.g., [48]).

Early work on the modelling and analysis of MAPE-K feedback loops essentially treated the knowledge base as a shared memory between processes [2, 23].

However, in DTs the knowledge base tends to capture domain knowledge as a static context, for example using knowledge graphs [22]. In this setting, we may understand the knowledge base of a self-adaptive system such as a DT as a system invariant; i.e., a set of properties that should hold for the system in quiescent states during execution. However, the knowledge base of a DT can be rather complex, especially if it includes contextual information such as domain knowledge.

Challenge 4 *How can formal methods improve guarantees for the expected behaviour and services of a digital twin?*

While *consistency* between different models and components is a common research topic in DT research [45], the notion of *correctness*, especially functional correctness, has not been investigated in depth. We take the view that correctness for DTs needs to relate the twin's configuration and behaviour to that of the actual system. However, we cannot generally assume that the actual system is fully understood; instead it is observed through its sensor readings and interpreted in the context of the domain knowledge. If we assume that the knowledge base is formally represented as a set of logical formulae, correctness can be expressed as a relation between the DT's Layer 1 and the knowledge base. Seen as an invariant, the knowledge base should have the current state of the DT's Layer 1 as a model. In our work, we have approached this problem through *semantic lifting* [32], a technique to integrate the runtime state of a DT in a knowledge base [33]. By providing a specific structure to the knowledge base and to the operations used for self-adaptation, both correctness and quiescent states can then be formally expressed (see, e.g., [50]).

Challenge 5 *How can we reason about systems that interact with external knowledge bases?*

In fact, little work has been done on assessing the behaviour of software that interacts with external knowledge. We have opened a line of work on testing such software, using mutations over knowledge graphs to evaluate the robustness of the software using the knowledge and identifying the assumptions over the structure of the graph that are crucial for the software [25–27]. In a similar vein, we have shown how fuzzing techniques can be used to find bugs in software for interacting with knowledge graphs [24], such as logical reasoners. One may think of these assumptions as an interface between the software and the external knowledge, and we have introduced a verification system that integrates information from the knowledge base directly into a Hoare logic [30]. Complementing this line of work, Dubslaff et al. have developed an approach to incorporate domain knowledge in the form of ontologies into model checking [14].

Challenge 6 *How can we provide programmatic support for self-adaptation with external knowledge bases?*

While most work on self-adaptive systems fall into the "systems" category of computer science research, it is interesting to see if more programmatic support

can be provided for developing DTs and related software. We have explored this problem by proposing *semantically reflected programs* in SMOL [31,32]. SMOL is a programming language specially constructed for programs that interact with an external knowledge base. The runtime states of SMOL can be automatically integrated in this knowledge base, to support introspection: programs can query the knowledge base about how their runtime state relates to external knowledge, and make runtime decisions based on that. In this way, SMOL is designed to support DTs that use external knowledge bases [34,35]. The integration of mechanisms for dynamic reclassification of objects in SMOL provides programmatic support for type-safe self-adaptation [50].

5 Conclusion

This paper argues why the emergence of DTs should be of interest to the formal methods community. We claim that DTs give us an opportunity to revisit the relationship between models and modelled systems, and explore how data-driven and self-adaptive aspects of DTs require a dynamic way of developing and composing models to perform on-the-fly analyses. The paper considers both the use of formal methods to deliver analysis services inside a DT and the use of formal methods to analyse the DT itself, and suggests a number of interesting, open challenges for model development, composition and integration in the context of DTs. These challenges touch upon model heterogeneity, correctness for self-adaptive systems such as DTs, and interaction with external knowledge such as formalised domain expertise. Admittedly, we are barely scratching the surface by indicating these specific challenges, with a focus on methodology, at the intersection of formal methods and DTs. We have concretised each challenge by indicating a direction of research in our own work. We believe there is a huge potential here that deserves more attention from the formal methods community.

References

1. Ahlgren, J., Bojarczuk, K., Drossopoulou, S., Dvortsova, I., George, J., Gucevska, N., Harman, M., Lomeli, M., Lucas, S.M.M., Meijer, E., Omohundro, S., Rojas, R., Sapora, S., Zhou, N.: Facebook's cyber-cyber and cyber-physical digital twins. In: Proc. Evaluation and Assessment in Software Engineering (EASE 2021). pp. 1–9. ACM (2021). https://doi.org/10.1145/3463274.3463275
2. Arcaini, P., Riccobene, E., Scandurra, P.: Modeling and analyzing MAPE-K feedback loops for self-adaptation. In: Inverardi, P., Schmerl, B.R. (eds.) Proc. 10th Intl. Symp. on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2015). pp. 13–23. IEEE Computer Society (2015). https://doi.org/10.1109/SEAMS.2015.10
3. Baier, C., de Alfaro, L., Forejt, V., Kwiatkowska, M.: Model checking probabilistic systems. In: Handbook of Model Checking, pp. 963–999. Springer (2018). https://doi.org/10.1007/978-3-319-10575-8_28
4. Baresi, L., Ghezzi, C.: The disappearing boundary between development-time and run-time. In: Proc. Workshop on Future of Software Engineering Research, (FoSER 2010). pp. 17–22. ACM (2010). https://doi.org/10.1145/1882362.1882367

5. ter Beek, M.H., Chapman, R., Cleaveland, R., Garavel, H., Gu, R., ter Horst, I., Keiren, J.J.A., Lecomte, T., Leuschel, M., Rozier, K.Y., Sampaio, A., Seceleanu, C., Thomas, M., Willemse, T.A.C., Zhang, L.: Formal Methods in Industry. Formal Aspects Comput. **37**(1), 7:1–7:38 (2025). https://doi.org/10.1145/3689374
6. Billey, A., Wuest, T.: Energy digital twins in smart manufacturing systems: A case study. Robotics Comput. Integr. Manuf. **88**, 102729 (2024). https://doi.org/10.1016/J.RCIM.2024.102729
7. Birchler, C., Khatiri, S., Rani, P., Kehrer, T., Panichella, S.: A roadmap for simulation-based testing of autonomous cyber-physical systems: Challenges and future direction. ACM Trans. Softw. Eng. Methodol. **34**(5), 152:1–152:9 (2025). https://doi.org/10.1145/3711906
8. Bjørner, N.S., Phan, A., Fleckenstein, L.: νz - an optimizing SMT solver. In: Proc. TACAS 2015. LNCS, vol. 9035, pp. 194–199. Springer (2015). https://doi.org/10.1007/978-3-662-46681-0_14
9. Blair, G.S., Bencomo, N., France, R.B.: Models@ run.time. Computer **42**(10), 22–27 (2009). https://doi.org/10.1109/MC.2009.326
10. Braun, S., Dalibor, M., Jansen, N., Jarke, M., Koren, I., Quix, C., Rumpe, B., Wimmer, M., Wortmann, A.: Engineering digital twins and digital shadows as key enablers for industry 4.0. In: Digital Transformation - Core Technologies and Emerging Topics from a Computer Science Perspective, pp. 3–31. Springer (2022). https://doi.org/10.1007/978-3-662-65004-2_1
11. Chang, X., Zhang, R., Mao, J., Fu, Y.: Digital twins in transportation infrastructure: An investigation of the key enabling technologies, applications, and challenges. IEEE Trans. Intell. Transp. Syst. **25**(7), 6449–6471 (2024). https://doi.org/10.1109/TITS.2024.3401716
12. Chrszon, P., Dubslaff, C., Klüppelholz, S., Baier, C.: ProFeat: Feature-Oriented Engineering for Family-Based Probabilistic Model Checking. Formal Aspects Comput. **30**(1), 45–75 (2018). https://doi.org/10.1007/s00165-017-0432-4
13. Classen, A., Cordy, M., Schobbens, P.Y., Heymans, P., Legay, A., Raskin, J.F.: Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking. IEEE Transaction on Software Engineering **39**(8), 1069–1089 (2013). https://doi.org/10.1109/TSE.2012.86
14. Dubslaff, C., Koopmann, P., Turhan, A.: Enhancing probabilistic model checking with ontologies. Formal Aspects Comput. **33**(6), 885–921 (2021). https://doi.org/10.1007/S00165-021-00549-0
15. Fitzgerald, J., Gomes, C., Larsen, P.G. (eds.): The Engineering of Digital Twins. Springer (2024). https://doi.org/10.1007/978-3-031-66719-0
16. Gleirscher, M., Marmsoler, D.: Formal Methods in Dependable Systems Engineering: A Survey of Professionals from Europe and North America. Empir. Softw. Eng. **25**(6), 4473–4546 (2020). https://doi.org/10.1007/s10664-020-09836-5
17. Gomes, C., Thule, C., Broman, D., Larsen, P.G., Vangheluwe, H.: Co-simulation: A survey. ACM Comput. Surv. **51**(3), 49:1–49:33 (2018). https://doi.org/10.1145/3179993
18. Hallsteinsen, S., Hinchey, M., Park, S., Schmid, K.: Dynamic Software Product Lines. In: Systems and Software Variability Management: Concepts, Tools and Experiences, pp. 253–260. Springer (2013). https://doi.org/10.1007/978-3-642-36583-6_16
19. Hansen, S.T., Kamburjan, E., Kazemi, Z.: Monitoring reconfigurable simulation scenarios in co-simulated digital twins. In: Proc. 12th Intl. Symp. on Leveraging Applications of Formal Methods, Verification and Validation. Application Areas

(ISoLA 2024). LNCS, vol. 15223, pp. 47–61. Springer (2024). https://doi.org/10.1007/978-3-031-75390-9_4
20. Hierons, R.M., Bogdanov, K., Bowen, J.P., Cleaveland, R., Derrick, J., Dick, J., Gheorghe, M., Harman, M., Kapoor, K., Krause, P.J., Lüttgen, G., Simons, A.J.H., Vilkomir, S.A., Woodward, M.R., Zedan, H.: Using formal specifications to support testing. ACM Comput. Surv. **41**(2), 9:1–9:76 (2009). https://doi.org/10.1145/1459352.1459354
21. Hinchey, M., Park, S., Schmid, K.: Building Dynamic Software Product Lines. IEEE Computer **45**(10), 22–26 (2012). https://doi.org/10.1109/MC.2012.332
22. Hogan, A., et al.: Knowledge graphs. ACM Comput. Surv. **54**(4) (Jul 2021). https://doi.org/10.1145/3447772
23. de la Iglesia, D.G., Weyns, D.: MAPE-K formal templates to rigorously design behaviors for self-adaptive systems. ACM Trans. Auton. Adapt. Syst. **10**(3), 15:1–15:31 (2015). https://doi.org/10.1145/2724719
24. John, T., Johnsen, E.B., Kamburjan, E., Steinhöfel, D.: Language-based testing for knowledge graphs. In: Proc. 22nd European Semantic Web Conf. (ESWC 2025). LNCS, vol. 15719, pp. 24–46. Springer (2025). https://doi.org/10.1007/978-3-031-94578-6_2
25. John, T., Kamburjan, E., Johnsen, E.B.: Mutation-based integration testing of knowledge graph applications. In: Proc. 35th Intl. Symp. on Software Reliability Engineering (ISSRE 2024). pp. 475–486. IEEE (2024). https://doi.org/10.1109/ISSRE62328.2024.00052
26. John, T., Kamburjan, E., Johnsen, E.B.: Mutation-based testing of knowledge graphs. Empir. Softw. Eng. (2026), to appear.
27. John, T., Kamburjan, E., Johnsen, E.B.: RDFMutate: Mutation-based generation of knowledge graphs. In: Proc. 24th Intl. Semantic Web Conf. (ISWC 2025). LNCS, vol. 16141, pp. 295–312. Springer (2026). https://doi.org/10.1007/978-3-032-09530-5_17
28. Johnsen, E.B., Hähnle, R., Schäfer, J., Schlatte, R., Steffen, M.: ABS: A core language for abstract behavioral specification. In: Proc. FMCO. LNCS, vol. 6957, pp. 142–164. Springer (2010). https://doi.org/10.1007/978-3-642-25271-6_8
29. Kamburjan, E., Bencomo, N., Tapia Tarifa, S.L., Johnsen, E.B.: Declarative lifecycle management in digital twins. In: Proc. 1st Intl. Conf. on Engineering Digital Twins (EDTconf 2024). pp. 353—-363. MODELS Companion'24, ACM (2024). https://doi.org/10.1145/3652620.3688248
30. Kamburjan, E., Gurov, D.: Multi-perspective correctness of programs. In: Proc. 22nd Intl. Colloquium on Theoretical Aspects of Computing (ICTAC 2025). LNCS, vol. 16237, pp. 69–86. Springer (2025). https://doi.org/10.1007/978-3-032-11176-0_6
31. Kamburjan, E., Klungre, V.N., Qu, Y., Schlatte, R., Kostylev, E.V., Giese, M., Johnsen, E.B.: Semantically reflected programs. CoRR **abs/2509.03318** (2025). https://doi.org/10.48550/ARXIV.2509.03318
32. Kamburjan, E., Klungre, V.N., Schlatte, R., Johnsen, E.B., Giese, M.: Programming and debugging with semantically lifted states. In: Proc. 18th Extended Semantic Web Conference (ESWC 2021). LNCS, vol. 12731, pp. 126–142. Springer (2021). https://doi.org/10.1007/978-3-030-77385-4_8
33. Kamburjan, E., Pferscher, A., Schlatte, R., Sieve, R., Tapia Tarifa, S.L., Johnsen, E.B.: Semantic reflection and digital twins: A comprehensive overview. In: The Combined Power of Research, Education, and Dissemination: Essays Dedicated to Tiziana Margaria on the Occasion of Her 60th Birthday, LNCS, vol. 15240, pp. 129–145. Springer (2025). https://doi.org/10.1007/978-3-031-73887-6_11

34. Kamburjan, E., Sieve, R., Baramashetru, C.P., Amato, M., Barmina, G., Occhipinti, E., Johnsen, E.B.: GreenhouseDT: An exemplar for digital twins. In: Proc. 19th Intl. Symp. on Software Eng. for Adaptive and Self-Managing Systems (SEAMS 2024). pp. 175–181. ACM (2024). https://doi.org/10.1145/3643915.3644108
35. Kløvstad, Å.A.A., Kobialka, P., Sieve, R., Pferscher, A., Slaughter, L., Tapia Tarifa, S.L., Johnsen, E.B.: What-if scenarios for the BedreFlyt digital twin. In: Principles of formal quantitative analysis — Essays dedicated to Christel Baier on the occasion of her 60th birthday. p. 360–381. LNCS, Springer (2026). https://doi.org/10.1007/978-3-031-97439-7_18
36. Kobialka, P., Pferscher, A., Bergersen, G.R., Johnsen, E.B., Tapia Tarifa, S.L.: Stochastic games for user journeys. In: Proc. 26th Intl. Symp. on Formal Methods (FM 2024). LNCS, vol. 14934, pp. 167–186. Springer (2024). https://doi.org/10.1007/978-3-031-71177-0_12
37. Kobialka, P., Tapia Tarifa, S.L., Bergersen, G.R., Johnsen, E.B.: User journey games: automating user-centric analysis. Softw. Syst. Model. **23**(3), 605–624 (2024). https://doi.org/10.1007/S10270-024-01148-2
38. Korn, M.: Formal-Methods Support for Runtime Adaptation in Self-Adaptive Systems. Ph.D. thesis, Dresden University of Technology, Germany (2025)
39. Kristensen, M.H., Bonizzi, A., Gomes, C., Hansen, S.T., Martin, C.I.I., Iven, H., Kamburjan, E., Larsen, P.G., Leucker, M., Talasila, P., Tang, V.T., Tonetta, S., Vosteen, L.B., Wright, T.: Runtime verification of autonomous systems utilizing digital twins as a service. In: Proc. Intl. Conf. on Autonomic Computing and Self-Organizing Systems (ACSOS 2024). pp. 121–127. IEEE (2024). https://doi.org/10.1109/ACSOS-C63493.2024.00042
40. Kritzinger, W., Karner, M., Traar, G., Henjes, J., Sihn, W.: Digital twin in manufacturing: A categorical literature review and classification. IFAC-PapersOnLine **51**(11), 1016–1022 (2018). https://doi.org/10.1016/j.ifacol.2018.08.474
41. Kruger, L., Kobialka, P., Pferscher, A., Johnsen, E.B., Junges, S., Rot, J.: Incremental fingerprinting in an open world. In: Proc. CSF 2026. IEEE (2026). https://doi.org/10.48550/arXiv.2601.21680, to appear.
42. Laubenbacher, R., Mehrad, B., Shmulevich, I., Trayanova, N.: Digital twins in medicine. Nat Comput Sci **4**, 184–191 (2024). https://doi.org/10.1038/s43588-024-00607-6
43. Michael, J., et al.: Model-driven engineering for digital twins: Opportunities and challenges. Syst. Eng. **28**(5), 659–670 (2025). https://doi.org/10.1002/SYS.21815
44. Michalec, O.: Models vs infrastructures? on the role of digital twins' hype in anticipating the governance of the UK energy industry. Environmental Science & Policy **168**, 104041 (2025). https://doi.org/10.1016/j.envsci.2025.104041
45. Muctadir, H.M., Kamburjan, E., Cleophas, L., van den Brand, M.: A consistency management framework for digital twin models. Journal of Systems and Software **234**, 112750 (2026). https://doi.org/10.1016/j.jss.2025.112750
46. National Academies of Sciences, Engineering, and Medicine (NASEM): Foundational Research Gaps and Future Directions for Digital Twins. The National Academies Press (2024). https://doi.org/10.17226/26894
47. Pferscher, A., Wunderling, B., Aichernig, B.K., Muskardin, E.: Mining digital twins of a VPN server. In: Proc. Workshop on Applications of Formal Methods and Digital Twins. CEUR Workshop Proc. vol. 3507. CEUR-WS.org (2023), https://ceur-ws.org/Vol-3507/paper6.pdf

48. Päßler, J., ter Beek, M.H., Damiani, F., Dubslaff, C., Johnsen, E.B., Tapia Tarifa, S.L.: Feature-oriented modelling and analysis of a self-adaptive robotic system. Formal Aspects Comput. **37**(4), 1–39 (2025). https://doi.org/10.1145/3709159
49. Päßler, J., ter Beek, M.H., Damiani, F., Johnsen, E.B., Tapia Tarifa, S.L.: Analysing self-adaptive systems as software product lines. Journal of Systems and Software **222**, 112324 (2025). https://doi.org/10.1016/j.jss.2024.112324
50. Sieve, R., Kamburjan, E., Damiani, F., Johnsen, E.B.: Declarative dynamic object reclassification. In: Proc. ECOOP 2025. LIPIcs, vol. 333, pp. 29:1–29:31. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2025). https://doi.org/10.4230/LIPICS.ECOOP.2025.29
51. Sieve, R., Kobialka, P., Slaughter, L., Schlatte, R., Johnsen, E.B., Tapia Tarifa, S.L.: BedreFlyt: Improving patient flows through hospital wards with digital twins. In: Proc. First Intl. Workshop on Autonomous Systems Quality Assurance and Prediction with Digital Twins (ASQAP 2025). EPTCS, vol. 418, pp. 1–15. Open Publishing Association (2025). https://doi.org/10.4204/EPTCS.418.1
52. Torres, W., van den Brand, M.G.J., Serebrenik, A.: A systematic literature review of cross-domain model consistency checking by model management tools. Softw. Syst. Model. **20**(3), 897–916 (2021). https://doi.org/10.1007/S10270-020-00834-1
53. Vaandrager, F.W.: Model learning. Commun. ACM **60**(2), 86–95 (2017). https://doi.org/10.1145/2967606
54. Vachálek, J., Bartalský, L., Rovný, O., Šišmišová, D., Morháč, M., Lokšík, M.: The digital twin of an industrial production line within the Industry 4.0 concept. In: Proc. 21st Intl. Conf. on Process Control (PC 2017). pp. 258–262 (2017). https://doi.org/10.1109/PC.2017.7976223
55. Vallée, A.: Digital twin for healthcare systems. Frontiers in Digital Health **5**, 1253050 (Sep 2023). https://doi.org/10.3389/fdgth.2023.1253050
56. Wagenmaker, A., Huang, K., Ke, L., Jamieson, K.G., Gupta, A.: Overcoming the sim-to-real gap: Leveraging simulation to learn to explore for real-world RL. In: Proc. NeuRIPS 2024 (2024), http://papers.nips.cc/paper_files/paper/2024/hash/8fa068ffe59817175d176bd75641fe16-Abstract-Conference.html
57. Wallner, F., Aichernig, B.K., Burghard, C.: It's not a feature, it's a bug: Fault-tolerant model mining from noisy data. In: Proc. ICSE 2024. pp. 29:1–29:13. ACM (2024). https://doi.org/10.1145/3597503.3623346
58. Weyns, D.: An Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective. Wiley-IEEE Computer Society Pr (2021). https://doi.org/10.1002/9781119574910
59. Weyns, D., Iftikhar, M.U., de la Iglesia, D.G., Ahmad, T.: A survey of formal methods in self-adaptive systems. In: Proc. Fifth Intl. C* Conf. on Computer Science & Software Engineering (C3S2E 2012). pp. 67–79. ACM (2012). https://doi.org/10.1145/2347583.2347592
60. Wing, J.M.: A specifier's introduction to formal methods. Computer **23**(9), 8–24 (1990). https://doi.org/10.1109/2.58215
61. Woodcock, J., Larsen, P.G., Bicarregui, J., Fitzgerald, J.: Formal methods: Practice and experience. ACM Comput. Surv. **41**(4), 19:1–19:36 (2009). https://doi.org/10.1145/1592434.1592436
62. Zhao, W., Queralta, J.P., Westerlund, T.: Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In: Proc. Symposium Series on Computational Intelligence (SSCI 2020). pp. 737–744. IEEE (2020). https://doi.org/10.1109/SSCI47803.2020.9308468