

UiO : **Department of Informatics**  
University of Oslo

Preproceedings of the  
Workshop on Applications of  
Formal Methods and Digital Twins

Co-located with FM'23 in Lübeck, Germany

*Report 505*

*Editors: Eduard Kamburjan, Stefan Hallerstedde*

ISBN: 978-82-7368-605-3

ISSN: 0806-3036





This technical reports contains the papers of the Workshop on Applications of Formal Methods and Digital Twins, which is held 06.03.23 in Lübeck, Germany, co-located with the 25th International Symposium on Formal Methods. All papers were peer-reviewed prior to inclusion in this volume by the program committee of the workshop, and we thank the member of the PC for their constructive reviews. Additionally to the papers, the workshop will feature an invited talk by Ana Cavalcanti title “*RoboStar Twins?*”.

- *Leveraging Runtime Verification for the Monitoring of Digital Twins*  
by Sylvain Hallé, Chukri Soueidi and Yliès Falcone
- *Emerging Challenges in Compositionality and Correctness for Digital Twins*  
by Eduard Kamburjan, Vidar Klungre, Silvia Lizeth Tapia Tarifa, Rudolf Schlatte, Martin Giese, David Cameron and Einar Broch Johnsen
- *Are Formal Contracts a useful Digital Twin of Software Systems?*  
by Jonas Schiffel and Alexander Weigl
- *Digital Twin for Rescue Missions – a Case Study*  
by Martin Leucker, Martin Sachenbacher and Lars Bernd Vosteen
- *A Digital Twin for Coupling Mobility and Energy Optimization: The ReNuBiL Living Lab*  
by Daniel Thoma, Martin Sachenbacher, Martin Leucker and Aliyu Tanko Ali
- *Mining Digital Twins of a VPN Server*  
by Andrea Pferscher, Benjamin Wunderling, Bernhard K. Aichernig and Edi Muskardin

# Leveraging Runtime Verification for the Monitoring of Digital Twins

Sylvain Hallé<sup>1</sup>, Chukri Soueidi<sup>2</sup>, and Yliès Falcone<sup>2</sup>

<sup>1</sup> Laboratoire d’informatique formelle  
Université du Québec à Chicoutimi, Canada

<sup>2</sup> Laboratoire d’informatique de Grenoble  
Université Grenoble Alpes, France

**Abstract.** The paper considers the problem of discovering divergences between the actions of a digital twin and those of its real-world counterpart. It observes the similarities between this problem and an existing field of formal methods called Runtime Verification (RV), and suggests leveraging and adapting RV techniques to this effect. Concretely, three important aspects of the problem are identified and for which both theoretical and practical challenges must be addressed.

## 1 Introduction

A *digital twin* is a virtual representation of a real-world entity [13]; it is often presented as a predictive instrument, by enabling one to simulate multiple possible outcomes of a real-world entity. In such a context, it is essential that the digital twin exhibits behavior that is faithful to that of the system it seeks to mimic. Any significant and sustained discrepancy between the twin and its concrete counterpart can lead to incorrect predictions, false diagnoses, and generally to an incorrect perception of the operation of the real system. Differences between the operation of a twin and the real-world entity must therefore be monitored and addressed in real time as they occur.

The process of detecting deviations can be summarized as illustrated in Figure 1. A real-world entity  $E$  is given inputs  $I$ , which can consist of controllable (i.e. user-defined) values, as well as uncontrollable (i.e. environmental) objects, in addition to any reading related to the entity’s current internal state. The entity reacts to these inputs by producing outputs  $O_E$ ; again, these outputs can be actions directly performed by the entity, or measured values of the entity’s state or the environment. In parallel, the inputs are recorded and fed to a digital twin  $T$ , which simulates the real-world entity and produces its own outputs  $O_T$ . The observed output and the synthetic output are then fed to a comparison procedure  $C$ , which decides whether they are in agreement ( $\top$ ) or not ( $\perp$ ).

In this paper, we reveal the similarities that this problem shares with a research topic crossing the fields of software engineering and formal methods, called *runtime verification* (RV) [3]. Over the years, RV has been successfully applied to various use cases, ranging from the monitoring of aerial drones [23] to

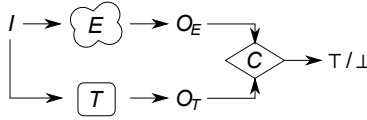


Fig. 1: Comparing the behavior of a real-world entity with that of a digital twin.

the detection of bugs in video games [28]. We identify elements of the problem that require adaptations in order to leverage RV for digital twin monitoring, and suggest possible ways in which these techniques could be used in this context.

## 2 A Property-Based Approach

Runtime verification is the discipline of computer science where an object called a *monitor* is used to observe the behavior of another program. At various moments, instrumented codepoints relay information about the program’s actions and state to the monitor, producing a sequence of data elements called “events”, denoted by  $\bar{\sigma} = \sigma_1, \sigma_2, \dots$ . The monitor compares this event trace against a formal specification  $\varphi$  of what constitutes a correct execution; any violation of the specification is reported on-the-fly, as the monitored program executes. Research on runtime verification has produced a variety of monitors supporting a large class of specification languages [5, 7, 10, 22, 25].

*A Passive Operation* As in RV, the monitoring of digital twins is mostly a *passive* procedure: one is not concerned with generating inputs that drive the system, as is the case for conformance testing [8, 20]; this is typically done by calculating the sequences of inputs that have the potential to reveal a violation of the finite-state machine specification by the system. However, in the context of digital twins, one rarely has the possibility of sending unlimited sequences of inputs to the concrete entity to ensure the conformity of the model, due to their associated cost (in terms of time and resources). What is more, some scenarios may involve compliance checking in situations where the actual system is damaged, and may be excluded from the test cases from the outset for this reason. Realistically, the best that can be done is to observe the behavior of the actual system in its normal operation, and to make the most of these observations to detect any discrepancies as they occur.

*Properties on Digital Twins* A first possible application of runtime verification consists of an indirect comparison between the twin’s behavior and that of the real-world entity. It supposes the existence of conditions  $\varphi_1, \dots, \varphi_n$ , which are known to be true for all executions of the digital twin; these properties, acting as a form of “guarantee,” are extracted from the twin beforehand by an arbitrary procedure  $G$ , as illustrated in Figure 2. These properties, in turn, can be converted into monitors that observe the operation of the real-world entity in real time. Any observed sequence that violates one of the  $\varphi_i$  is, by definition, a sequence that



cannot be produced by the digital twin for the same inputs, and thus indicates a divergence between the twin and its real-world counterpart.

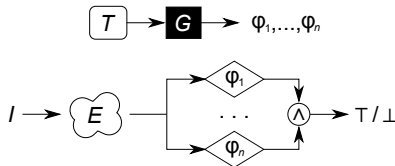


Fig. 2: Using properties on input/output sequences to detect divergences in the execution of a twin.

Note that in such a scenario, the twin itself does not need to be “executed”—that is, it is unnecessary to have the twin generate one or more traces for comparison with the real-world entity’s output. This approach can present advantages in cases where running a twin may incur a high cost. Instead of a constant synchronization between the twin and the real-world entity, the lighter monitors may observe the output of the entity as long as it corresponds to expected behavior. Further analysis (and possible adaptation of the twin) is only required when one of the properties is violated by the observations. This, in itself, may require substantial analysis to determine which are the relevant  $\varphi_i$  that need to be monitored.

*Declarative Definition of a Twin* In this first suggested mode of operation, the detection of divergences is *sound*: a violation of one of the properties indicates a genuine discrepancy between the twin and the real-world entity. The detection is *complete* if any sequence satisfying  $\bigwedge \varphi_i$  is also a possible (valid) sequence produced by the twin. In such a case, the twin’s behavior is completely captured by the conditions  $\varphi_i$  that serve as definitions. Thus, one can imagine specifying the operation of the twin in a *declarative* way, as opposed to a “procedural” or “imperative” way. Instead of programming the twin as an executable object that receives inputs and produces outputs, the execution of the twin is driven by a satisfiability (SAT) solver: given a sequence of inputs, the solver finds a sequence of outputs satisfying the properties, and returns that sequence as the twin’s reaction to the inputs. Such a declarative approach has already been attempted to simulate the execution of web services from temporal logic specifications [14].

### 3 Generalizing Runtime Verification

This proposed approach is the most direct application of runtime verification to digital twins. However, its soundness rests on the hypothesis that a set of properties of the twin can be extracted and used as formal guarantees on its execution. For a twin that is defined procedurally, those properties can be

deduced from its implementation (for example by defining properties manually and verifying them through model checking [2], or by observing multiple executions of the twin and deducing a formal model of its execution using process mining [27]).

However, obtaining these guarantees may be a complex process, and completely capturing the behavior of the twin in such a way may not be a realistic assumption. Another possibility consists of handling the twin as a black box, and to directly compare its output to that of the real-world entity (for a given input sequence), as described in Figure 1. A second point that this article puts forward is that this comparison can be framed as a generalization of runtime verification.

*Monitoring Two Traces* Contrary to RV, monitoring twins involves not one, but (at least) two traces at the same time ( $O_E$  and  $O_T$ ). The “property” that needs to be evaluated in such a case correlates the events observed in both traces. This is a particular case of what is called a *hyperproperty* [6, 12]; while a property determines whether a trace is valid or not in isolation, in hyperproperties traces are valid or not based on their relation with other traces. It shall be noted that this operation, taken in its most abstract form, can be any calculation; as we shall discuss below, it is not restricted to the pairwise comparison of events at matching indices in both traces, and can involve arbitrary constraints on the values and ordering of events at various locations.

*Expressiveness* As a matter of fact, an anticipated challenge for the leveraging of RV techniques to digital twin monitoring is the relatively low expressiveness of the notations they use as their specification language. A recent taxonomy of existing RV approaches highlights the fact that many of them use formalisms based on propositional temporal logic or finite-state automata [11]. Some of them have support for quantification over data values, or implement basic forms of aggregation such as sum or average [5, 9, 10, 16]. However, these languages are ill-suited in their present form to handle the rich event types and complex (and often numerical) relationships that can involve the values they contain —and most importantly, specification languages for hyperproperties are even more restricted than for classical properties.

*Towards Stream Processing* In this context, it might be desirable to expand the vision of the problem and to consider it as a more general form of complex event processing (CEP) [1, 17]. CEP is typically concerned with data-rich events, and considers arbitrary calculations over these events in order to produce higher-level (i.e. “complex”) events. Its more general computational model could be harnessed in order to express the potentially intricate operations required for uncovering discrepancies between the output of a digital twin and that of its real-world counterpart. Previous works have shown how CEP engines can evaluate properties specified in many languages used in RV and extend them with additional constructs [15], making them good candidates to address the expressiveness issues mentioned above.

## 4 Qualifying Divergence

Another important challenge lies in the fact that not all divergences between an entity and its twin are meaningful and indicative of a problem. There are situations where discrepancies between an entity and its twin are expected, if not unavoidable, and still do not represent any malfunction, modeling error, or significant drift between a twin and its real-world counterpart.

*Sources of Divergence* A first such case is caused by *uncertainty* in measurements. A physical entity will typically have its environment and behavior measured by sensors, whose output is intrinsically tainted with uncertainty, bias, or even spurious drop-outs. Thus one cannot expect a strict equality between values produced (or predicted) by a twin and those measured in a real-world system. It turns out that several works in the field of RV have addressed the issue of verifying properties in the presence of missing or imprecise events [4,18,26], which could be leveraged to the context of digital twins.

A second source of divergence may be caused by different *interleavings* in the events produced by a twin and its counterpart. This can happen when events produced by multiple components of the system happen more or less simultaneously, and are arbitrarily flattened to a particular ordering which may differ depending on uncontrollable or external factors. Instrumentation is sometimes designed on purpose to lose some ordering information, as is the case in some cyber-physical systems [29].

A final source of divergences comes from *under-specification*. In some cases, it is expected that the digital twin is run from a coarse-grained description of the real-world entity that does not totally capture its internal state. This typically shows up as the system appearing to operate non-deterministically, producing different outputs in what are apparently identical input conditions. In some other cases, non-determinism may simply arise from the fact that multiple possible (and equally acceptable) outcomes are possible for the same set of initial conditions.

For all these reasons, it is unrealistic that the comparison procedure  $C$  shown in Figure 1 looks at  $O_E$  and  $O_T$  and simply expects both to be identical. We identify two ways of dealing with this issue, one being the opposite of the other, and illustrated in Figure 3.

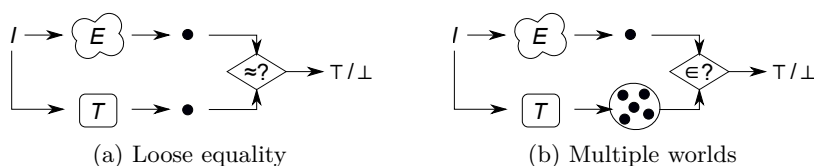


Fig. 3: Two possible ways of comparing the output of a twin with that of a real-world system while tolerating some divergence.



*Single World, Loose Equality* A first possibility, illustrated in Figure 3a is to let a digital twin produce a unique output for a given input (i.e. a “single possible world”). However, it is allowed (and even expected) that this output  $O_T$  differs slightly from  $O_E$ ; therefore, the comparison procedure  $C$  does not look for strict equality between the two streams, but rather evaluates a relaxed property. For example, for a system producing a stream of numerical values, one may expect that the running average over a sliding window of  $k$  values is the same, but not the individual events produced by both systems. In mathematical terms, the comparison criterion is an equivalence relation that is looser than equality. For numerical values, this can be likened to fuzzy comparators [19].

This mode of operation brings challenges of its own. First, an appropriate equivalence relation must be defined, and it is expected that such relation be specific to each problem domain. Second is the necessity of evaluating this relation efficiently at runtime, over two streams of events that are progressively produced by both the twin and the real-world entity. Deviations should be reported on-the-fly, as obviously one cannot wait for the executions to complete before starting the comparison. This seemingly innocuous observation makes it difficult to use well-known string distance criteria, such as Levenshtein distance [24], which have a high computational complexity and typically expect strings to be completely known in advance to calculate their value.

*Multiple Worlds, Strict Equality* An alternate approach, illustrated in Figure 3b, is to allow a twin to produce multiple possible outputs for a given input. Each of these outputs corresponds to one of the “possible worlds” the system can be in for given input conditions. Strict equality is then sought between the real-world entity’s output, and one of those possible worlds<sup>3</sup>. In this setting, the multiple possible worlds can either be represented explicitly (as an enumeration of all possible outputs) or implicitly (through an abstract property that it satisfied by all possible worlds). For example, a numerical value associated with a precision interval counts as an abstract representation of multiple exact numerical values.

The multi-world trace semantics has been used to address the question of runtime verification with uncertainty [21,26], and could be adapted to the problem of stream comparison for digital twins.

## 5 Conclusion

In this paper, we have highlighted the connections that can be made between the question of detecting discrepancies between a digital twin and a concrete entity, and the problem of runtime verification already studied in the community of software engineering and formal methods. Although they present clear similarities, these two problems are nevertheless distinct, and some adaptation is therefore necessary in order to leverage existing runtime verification techniques to the particular context of digital twins.

<sup>3</sup> Stated otherwise, the entity’s output must be included in the possible outputs produced by the twin.

The article identified several research directions aimed at enabling real-time divergence checking using VR techniques, which will be explored in more detail in future work. Among these, we note the design of specification languages that are more expressive and appropriate to the problem of digital twins, as well as the definition of trace comparison metrics that are less strict than a simple position-by-position equality.

## References

1. Event stream processing (ESP). In: Liu, L., Özsu, M.T. (eds.) *Encyclopedia of Database Systems*, p. 1064. Springer US (2009)
2. Baier, C., Katoen, J.P.: *Principles of Model Checking*. MIT Press (2008)
3. Bartocci, E., Falcone, Y., Francalanza, A., Reger, G.: Introduction to runtime verification. In: Bartocci, E., Falcone, Y. (eds.) *Lectures on Runtime Verification - Introductory and Advanced Topics*, *Lecture Notes in Computer Science*, vol. 10457, pp. 1–33. Springer (2018)
4. Basin, D.A., Klaedtke, F., Marinovic, S., Zalinescu, E.: Monitoring compliance policies over incomplete and disagreeing logs. In: Qadeer, S., Tasiran, S. (eds.) *RV. Lecture Notes in Computer Science*, vol. 7687, pp. 151–167. Springer (2012)
5. Basin, D.A., Klaedtke, F., Marinovic, S., Zalinescu, E.: Monitoring of temporal first-order properties with aggregations. *Formal Methods Syst. Des.* **46**(3), 262–285 (2015)
6. Clarkson, M.R., Schneider, F.B.: Hyperproperties. In: *CSF*. pp. 51–65. IEEE Computer Society (2008)
7. Colombo, C., Pace, G.J., Schneider, G.: LARVA — safer monitoring of real-time Java programs (tool paper). In: Hung, D.V., Krishnan, P. (eds.) *SEFM*. pp. 33–37. IEEE Computer Society (2009)
8. Constant, C., Jéron, T., Marchand, H., Rusu, V.: Integrating formal verification and conformance testing for reactive systems. *IEEE Trans. Software Eng.* **33**(8), 558–574 (2007)
9. Convent, L., Hungerecker, S., Leucker, M., Scheffel, T., Schmitz, M., Thoma, D.: TeSSLa: Temporal stream-based specification language. In: Massoni, T., Mousavi, M.R. (eds.) *SBMF. Lecture Notes in Computer Science*, vol. 11254, pp. 144–162. Springer (2018)
10. D’Angelo, B., Sankaranarayanan, S., Sánchez, C., Robinson, W., Finkbeiner, B., Sipma, H.B., Mehrotra, S., Manna, Z.: LOLA: runtime monitoring of synchronous systems. In: *TIME*. pp. 166–174. IEEE Computer Society (2005)
11. Falcone, Y., Krstic, S., Reger, G., Traytel, D.: A taxonomy for classifying runtime verification tools. *Int. J. Softw. Tools Technol. Transf.* **23**(2), 255–284 (2021)
12. Finkbeiner, B., Hahn, C., Stenger, M., Tentrup, L.: Monitoring hyperproperties. *Formal Methods Syst. Des.* **54**(3), 336–363 (2019)
13. Grieves, M., Vickers, J.: Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems, pp. 85–113. Springer (2017)
14. Hallé, S.: Model-based simulation of SOAP web services from temporal logic specifications. In: Perseil, I., Breitman, K.K., Sterritt, R. (eds.) *ICECCS*. pp. 95–104. IEEE Computer Society (2011)
15. Hallé, S., Khoury, R.: Writing domain-specific languages for beepbeep. In: Colombo, C., Leucker, M. (eds.) *Runtime Verification - 18th International Conference, RV 2018, Limassol, Cyprus, November 10-13, 2018, Proceedings. Lecture Notes in Computer*

- Science, vol. 11237, pp. 447–457. Springer (2018). [https://doi.org/10.1007/978-3-030-03769-7\\_27](https://doi.org/10.1007/978-3-030-03769-7_27), [https://doi.org/10.1007/978-3-030-03769-7\\_27](https://doi.org/10.1007/978-3-030-03769-7_27)
16. Hallé, S., Villemaire, R.: Runtime enforcement of web service message contracts with data. *IEEE Trans. Serv. Comput.* **5**(2), 192–206 (2012)
  17. Hallé, S.: *Event Stream Processing With BeepBeep 3: Log Crunching and Analysis Made Easy*. Presses de l’Université du Québec (2018)
  18. Joshi, Y., Tchamgoue, G.M., Fischmeister, S.: Runtime verification of LTL on lossy traces. In: Seffah, A., Penzenstadler, B., Alves, C., Peng, X. (eds.) *SAC*. pp. 1379–1386. ACM (2017)
  19. Kaufmann, A., Gupta, M.M.: *Introduction to Fuzzy Arithmetic: Theory and Applications*. Van Nostrand Reinhold (1991)
  20. Lee, D., Yannakakis, M.: Principles and methods of testing FSMs: A survey. *Proceedings of the IEEE* **84**, 1089–1123 (1996)
  21. Leucker, M., Sánchez, C., Scheffel, T., Schmitz, M., Thoma, D.: Runtime verification for timed event streams with partial information. In: Finkbeiner, B., Mariani, L. (eds.) *RV. Lecture Notes in Computer Science*, vol. 11757, pp. 273–291. Springer (2019)
  22. Meredith, P.O., Jin, D., Griffith, D., Chen, F., Rosu, G.: An overview of the MOP runtime verification framework. *Int. J. Softw. Tools Technol. Transf.* **14**(3), 249–289 (2012)
  23. Moosbrugger, P., Rozier, K.Y., Schumann, J.: R2U2: monitoring and diagnosis of security threats for unmanned aerial systems. *Formal Methods Syst. Des.* **51**(1), 31–61 (2017)
  24. Navarro, G.: A guided tour to approximate string matching. *ACM Comput. Surv.* **33**(1), 31–88 (2001)
  25. Reger, G., Cruz, H.C., Rydeheard, D.E.: Marq: Monitoring at runtime with QEA. In: Baier, C., Tinelli, C. (eds.) *TACAS. Lecture Notes in Computer Science*, vol. 9035, pp. 596–610. Springer (2015)
  26. Taleb, R., Khoury, R., Hallé, S.: Runtime verification under access restrictions. In: Bliudze, S., Gnesi, S., Plat, N., Semini, L. (eds.) *FormaliSE@ICSE*. pp. 31–41. IEEE (2021)
  27. van der Aalst, W.: *Process Mining: Data Science in Action*. Springer (2016)
  28. Varvaressos, S., Lavoie, K., Gaboury, S., Hallé, S.: Automated bug finding in video games: A case study for runtime monitoring. *Comput. Entertain.* **15**(1), 1:1–1:28 (2017)
  29. Wang, S., Ayoub, A., Sokolsky, O., Lee, I.: Runtime verification of traces under recording uncertainty. In: Khurshid, S., Sen, K. (eds.) *RV. Lecture Notes in Computer Science*, vol. 7186, pp. 442–456. Springer (2011)



# Emerging Challenges in Compositionality and Correctness for Digital Twins

Eduard Kamburjan, Vidar Norstein Klungre, S. Lizeth Tapia Tarifa, Rudolf Schlatte, Martin Giese, David Cameron, and Einar Broch Johnsen

Department of Informatics, University of Oslo, Oslo, Norway  
{`eduard,vidarkl,sltartifa,rudi,einarj,martingi,davidbc`}@ifi.uio.no

**Abstract.** A digital twin is an information system that analyzes the behavior of a physical or digital system by connecting streams of observations to dynamic (e.g., simulation) and static (e.g., asset management) models of this twinned system. In large-scale industrial settings, the digital twin will often need to manage a multitude of models for subsystems reflecting different engineering disciplines, vendors, etc. To analyze such complex systems, digital twins must ensure the correct composition of these models and their correct exposure to the user. For the integration and transfer of information between models, digital twins may profit from a formalization of domain knowledge using ontologies, which have proven effective to unify data models. However, it is an open challenge to formalize and verify the correctness of digital twins. This paper discusses this problem for digital twins and illustrates challenges for formal methods with a focus on the composition of heterogeneous dynamic models.

## 1 Introduction

Digital twins, originally conceived for NASA’s space programme [1], enable industry to significantly improve the *life-cycle management* of physical assets. The vision of digital twins is to create a digital replica (the “digital twin”), which is connected in real time to the modelled, traditionally cyber-physical, system (the “twinned system”). Via this real-time connection, the digital twin aims to provide insights into the twinned system’s state or behavior.

At the core of this vision, the digital twin coordinates data exchange between (a) the twinned system, (b) a range of model-based analysis tools and (c) stakeholders like engineers and analysts. The data about the twinned system typically combine static asset models and time-series measurements (e.g., data streams from sensors). The analysis tools typically combine simulators of physical models with executable software models. The digital twin computes an approximation of the behavior of the twinned system to explore “what-happened”, “what-may-happen” and “what-if” scenarios. The engineer can interact with the digital twin to access data, but also to perform more involved operations, e.g., to predict the consequences of changing system parameters, replacing components in the twinned system, or evaluate newly developed designs [2].

A digital twin is a composed, data-intensive system that needs to coordinate its analysis tools, data exchange between the twinned system and models that are

relevant for a particular analysis, as well as between different models if necessary. If the twinned system is physical, it consists of a *cyber-physical system* (CPS) in a *physical environment*, i.e., physical boundary conditions (e.g., temperature or fluid pressure) and modelled external actions (e.g., motion tracking devices). In the digital twin, both the CPS and the environment may be modeled by several components, each reflecting a part of the CPS or the dynamics of the operational environment. These smaller, targeted models are typically created by domain experts (e.g., chemical, mechanical or electrical engineers). Digital twins in industry are built from proprietary black-box applications, supplied by the vendor of the component. This limits the possibility to automate workflows within digital twins and to use formal tools to ensure basic correctness properties.

Nonetheless, digital twins are suited for formalization because of the inherent connection to model-based concepts. Challenges arise, besides black-box simulation, from the connection of complex data with complex dynamic models within the digital twin. Observe that there is a dichotomy between correctness for static and dynamic models: the integration of diverging static models can be achieved using semantic technologies, while the correct behavior and compositional constraints for dynamic models can be ensured using formal methods. A crucial step towards the formalization of digital twins is to connect these two approaches and formalize data propagation inside the twin.

While digital twins are often discussed from a data or business perspective [3], we take the formalization perspective in this paper to discuss the connection between static models of data and the composition of dynamic models. Correct data propagation between (and within) diverse models is related to orchestration in co-simulation [4], which is usually restricted to static structure. To configure a co-simulation system correctly for a particular analysis, different simulators need to be orchestrated to exchange data correctly. One particular approach to solve the challenge outlined above, is to combine **knowledge graphs** with orchestration in generalized co-simulation to ensure correctness. In this article we illustrate this, and further emerging challenges for formal methods with respect to integrating asset models and semantic technologies for digital twins.

*Related work.* *Semantically lifted programs* integrate static models represented using semantic technologies and dynamic models such as simulation units, into a programming language [5,6]. They have been applied to digital twins [7], but correctness has only been considered for specific applications [8,7]. Recent co-simulation surveys identify a lack of research into modular, stable, and accurate coupling of simulators in dynamic scenarios [4,9]. There is a long tradition to use semantic technologies to integrate data [10], in the digital twin context this is recently discussed [2,11,12].

## 2 Background

We briefly review the main concepts in co-simulation and ontologies, which form the basis for our discussion of digital twins and semantic technologies.

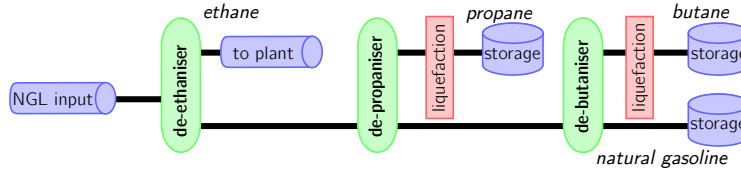


Fig. 1. Natural gas liquid fractionation plant, from [17].

*Co-simulation* denotes a way to implement global simulation of a complex system via the composition of various dynamic models representing the system’s components [4]. Each individual model, or *simulation unit*, can be seen as a black box capable of exhibiting behavior, consuming inputs and producing outputs. Assembling these simulation unit into a co-simulation poses some specific coordination challenges. The models must be synchronized not only wrt. the values they exchange (typically via point-to-point typed channels), but also on the current simulation time and when and by how much to advance time.

The time model of simulations, and hence co-simulations, can be *discrete* or *continuous*. In discrete event simulations, a simulation unit synchronizes with the environment at specific timestamps to exchange values. If two events happen at the same time, both are processed before the simulated time progresses. In continuous time simulation (e.g., for physical state), the state evolves continuously, which introduces flexibility in the step size of the time synchronization. For co-simulation scenarios which combine discrete and continuous parts, the orchestrator needs to reconcile the different assumptions about the inputs and outputs of each unit to retain the properties of the constituent systems.

*Semantic Technologies* [13] are techniques to formally attach meaning to data which can be used when constructing complex intelligent systems such as digital twins [12]. These techniques are based on *ontologies*: formal, conceptual descriptions of a domain, usually expressed in the *Web Ontology Language* (OWL) [14]. The ontology specifies the vocabulary of *classes* and *properties* that can be used by the system model, and a set of *axioms*, i.e., constraints, to which the model must adhere. Ontologies are used in many different domains, both within organizations, and as parts of large open projects, like SNOMED CT, an open ontology for clinical terms [15]. By introducing instances and combining them with classes and properties from the ontology, one can construct statements using the *resource description framework* (RDF) [16]; e.g., to model a concrete storage tank in some facility, one can assign an identifier (`:st1`) to the storage tank instance and connect it to the storage tank class (`:StorageTank`) given in the ontology: `:st1 a :StorageTank`. There is good tool support to check consistency of the resulting knowledge graphs (e.g., *do all axioms indeed hold?*), query them (e.g., *what are all the storage tanks?*), and reason over them to infer new facts, or check if concrete facts are implied.



### 3 A Simple Engineering Model

We use a small example to illustrate the challenges for coordinating models inside a digital twin. Fig. 1 shows the structure of a natural gas liquid (NGL) fractionation plant. Its input is natural gas liquids, or condensate, which is a mixture of light hydrocarbons (ethane, propane and butane). The purpose of the plant is to separate the light hydrocarbons. In the plant, the natural gas liquids are fed into distillation columns to isolate a single product: ethane, propane, then butane. Each column outputs two streams: a top product gas and the bottoms product that contains the remaining heavier hydrocarbons. The light gas products are either directly sent to a consumer (ethane is, e.g., used as a feed for petrochemicals plants or is burnt as fuel), or they are liquefied for sale.

Distillation is an expensive and energy-intensive process. Operating the plant requires us to monitor the fractionation process and determine optimal parameters like reflux rates and operating pressures for each distillation column and liquefaction unit. We can use dynamic models for simulation, based on non-linear systems of differential equations [18]. Model composition is constrained by domain knowledge about chemistry, thermodynamics and design practice. The parameters of the distillation and liquefaction units depend on the expected properties of the feed stock and constraints on the quality of the processed products. They are selected at design time to optimize the cost and performance of the plant. These parameters may be continuous variables (diameter of a column) or integers (number of trays in a distillation column).

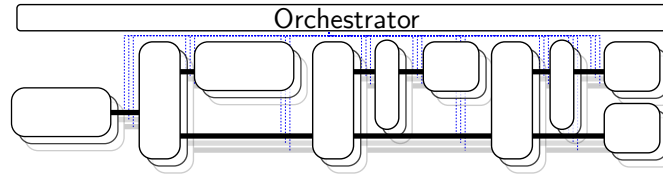
*Ontology.* An ontology for the fractionation plant can include two main classes: `:Pipeline` and `:Component`; each `:Component` must be either a `:Separator`, a `:StorageTank`, or a `:Liquefier`. Pipelines and components are connected by pipes, captured in the ontology with the object property `:isConnectedTo`. The part of the ontology concerned with separators and pipelines is then as follows:

```
:Component a owl:Class.      :Separator a owl:Class.
:Pipeline a owl:Class.       :isConnectedTo a owl:ObjectProperty.
```

Using this ontology, we construct the pipeline from the feed source (`:pipeline1`) and its connection to the de-ethaniser (`:separator1`) (see Fig. 1) as follows:

```
:pipeline1 a :Pipeline.        :separator1 a :Separator.
:pipeline1 :isConnectedTo      :separator1.
```

The model described here can be part of a digital twin, which additionally ensures correct data exchange and consistency both within the model *and* in its relation to the physical asset. (Remark that our terminology of a digital twin is sometimes called a digital twin architecture or a digital twin environment [19].) In particular, the digital twin must ensure correct data exchange not only between a dynamic model and a twinned system, but between different possible compositions of dynamic models, each running a different “what-if” scenario. These composed models cannot be used to control the twinned system, yet are connected to data streams from it, and possibly to the controlling model — it is paramount to keep explorative models connected to the controlling model, such that these do not influence the behavior of the twinned system directly.



**Fig. 2.** An orchestrator managing the connections of three composed dynamic models, based on Fig. 1; each box is a black-box simulator or a connection to a twinned asset.

## 4 Challenges

Development and formalization of digital twins beyond the industrial state of the art poses a number of challenges to the technologies employed. We identify two core challenges: Formalizing (a) the *correctness of digital twins*, and (b) the *principles of model composition* for a targeted physical or digital system.

Consider Fig. 2, which shows the structure of models that make up the digital twin. Note that the orchestrator need not be monolithic and that each of the boxes may be a dynamic model or a data stream from a twinned system. At each point in time, several dynamic model compositions may exist, with different configurations, for different purposes. Challenge (a) is to ensure that these composed dynamic models are internally consistent (i.e., they form meaningful co-simulation models), consistent with respect to the domain (i.e., they form models of some possible asset) and consistent with the actual twinned asset (i.e., the composed models and the twinned asset have the same structure).

*The Semantics of Composition.* Beyond checking for datatype violations and unconnected ports, the modeler must assess whether the composition of dynamic models is meaningful. Further checks are necessary, for example, whether the dynamic models are connected correctly with respect to the existing domain. In Fig. 2, the two output ports of each fractionation unit will have the same data type and physical unit (e.g., pressure or flow) but different semantic meanings; such consistency is a correctness property that relies on domain knowledge.

*Static and Dynamic Topologies.* The above challenge generalizes beyond connections: If the co-simulation is mirroring an asset (or asset model), then every meaningful component of the asset should be included in the co-simulation. Ensuring that the topology of the configuration is consistent with the domain must, again, take domain knowledge into account.

Observe that the notion of a digital replica touches on coordination aspects of self-organization [20], which must ensure that changing structure adheres to its domain constraints: The structure of the twinned system may change, e.g., due to planned maintenance (some components are shut off and exchanged) or unplanned repair. Tracking such changes is typically not supported by co-simulation frameworks or existing industrial practice such as [21], yet structural re-configuration is crucial in the digital twin to be able to use historical data without restarting the simulation system.

In our example, this corresponds to three scenarios: (a) Is the dynamic model indeed a replica of an existing system? (b) For a what-if analysis: is the modeled

system a possible fractionation plant? (c) For a maintenance analysis: does the proposed modification, adhere to the domain model? E.g., in the NGL example, if more information about `:separator1` and the connected tanks is available, we can use the representation of domain knowledge in an ontology to deduce whether the system adheres to the domain model.

*Coordinating Speculative Analyses.* The last challenges are concerned with one dynamic model, but as the digital twin moves from reproducing “what-happened” scenarios, in which the factual observations of the twinned system are known, to exploring possible “what-if” scenarios for its future behavior, the knowledge supplied by the twinned system decreases. E.g., one may want to explore how replacing a distillation column, or high environmental temperature, would affect the production of the plant as a whole. In these scenarios, there may be many solutions to the composition problem and the digital twin may need to speculatively explore and *coordinate* different possible solutions. Several of the composed models depicted in Fig. 2 may coexist; the composed models must share the connection to the twinned system and may also share some models.

*Asset Models.* The structural correctness of a dynamic model with respect to a twinned system requires that the twinned system already has a formal representation. One approach is to use *asset models* and *semantically lifted programs* to uniformly represent the twinned system and the dynamically composed model [7]. In particular for twinned physical systems, asset models can play a central role to achieve correctness and compositionality for digital twins: they formally describe requirements and topologies from the asset’s perspective, thereby providing the twin with static configuration data for model composition [22].

An asset model is an organized description of the composition and properties of an asset [23,24,25], used to support, e.g., maintenance operations on an asset. Asset models may be formalized as ontologies [26] or use them [27,28,29], with semantic data access being a current research focus [30,31,26]. We are in particular interested in *top-down asset models* which start by modeling the desired functionality of a system as a whole, and then decompose the system into functional sub-systems. This approach, which relates to model-driven engineering [32], is supported by modelling tools and languages such as SysML (e.g., [33]). A top-down model provides a scalable framework for tracking requirements along a system decomposition and linking requirements to individual components to higher-level system requirements [34,35]. We conjecture that top-down asset models can be used to tackle further challenges, by enriching them with information specific to digital twins.

## 5 Conclusion

Digital twins connect the management and development of a physical or digital system by applying analyses to a digital model in real time. In large-scale industrial settings, the asset is captured by a multitude of models, which stem from different engineering disciplines, different domain models and different vendors. Digital twins need to correctly integrate and exchange data between such

models. This paper discusses challenges for correctness and compositionality in the setting of digital twins, and proposes the use of asset models and formalized domain knowledge to enable formal methods to meet these challenges.

**Acknowledgements** This work was supported by the Norwegian Research Council via the SIRIUS Centre (237898) and the PeTWIN project (294600).

## References

1. Glaessgen, E., Stargel, D. In: The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles. AIAA (April 2012)
2. Cameron, D., Waaler, A., Komulainen, T.M.: Oil and gas digital twins after twenty years. How can they be made sustainable, maintainable and useful? In: SIMS 59. (2018)
3. Barricelli, B.R., Casiraghi, E., Fogli, D.: A survey on digital twin: Definitions, characteristics, applications, and design implications. IEEE Access **7** (2019)
4. Gomes, C., Thule, C., Broman, D., Larsen, P.G., Vangheluwe, H.: Co-simulation: A survey. ACM Comput. Surv. **51**(3) (2018)
5. Kamburjan, E., Johnsen, E.B.: *Knowledge Structures over Simulation Units*. In: ANNSIM, IEEE (2022)
6. Kamburjan, E., Klungre, V.N., Schlatte, R., Johnsen, E.B., Giese, M.: *Programming and Debugging with Semantically Lifted States*. In: ESWC. Volume 12731 of LNCS., Springer (2021)
7. Kamburjan, E., Klungre, V.N., Schlatte, R., Tapia, S.L.T., Cameron, D., Johnsen, E.B.: *Digital Twin Reconfiguration Using Asset Models*. In: ISoLA. Volume 13704 of LNCS., Springer (2022)
8. Kamburjan, E., Din, C.C., Schlatte, R., Tapia, S.L.T., Johnsen, E.B.: *Twining-by-Construction: Ensuring Correctness for Self-Adaptive Digital Twins*. In: ISoLA. Volume 13701 of LNCS., Springer (2022)
9. Schweiger, G., Gomes, C., Engel, G., Hafner, I., Schoeggl, J., Posch, A., Nouidui, T.S.: An empirical survey on co-simulation: Promising standards, challenges and research needs. Simul. Model. Pract. Theory **95** (2019)
10. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Ontology-based data access and integration. In Liu, L., Özsu, M.T., eds.: Encyclopedia of Database Systems. Springer New York, New York, NY (2018)
11. Rozanec, J.M., Jinzhi, L., Kosmerlj, A., Kenda, K., Dimitris, K., Jovanoski, V., Rupnik, J., Karlovcec, M., Fortuna, B.: Towards actionable cognitive digital twins for manufacturing. In: SeDiT@ESWC. Volume 2615 of CEUR., CEUR-WS (2020)
12. Kharlamov, E., Martín-Recuerda, F., Perry, B., Cameron, D., Fjellheim, R., Waaler, A.: Towards semantically enhanced digital twins. In: BigData, IEEE (2018)
13. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman and Hall/CRC Press (2010)
14. W3C, OWL Working Group: Web ontology language <https://www.w3.org/OWL>.
15. SNOMED International: SNOMED CT <https://www.snomed.org/>.
16. W3C, RDF Working Group: Resource description framework <https://www.w3.org/RDF>.
17. Jahn, F., Cook, M., Graham, M.: Hydrocarbon Exploration and Production. 2nd edn. Developments in Petroleum Science. Elsevier (2008)

18. Rahimi, A., Mustafa, M., Zaine, M., Ibrahim, N., Ibrahim, K., Yusoff, N., Al-Mutairi, E., Abd.Hamid, M.: Energy efficiency improvement in the natural gas liquids fractionation unit. *Chemical Engineering Transactions* **45** (2015)
19. Jones, D., Snider, C., Nassehi, A., Yon, J., Hicks, B.: Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology* **29** (2020)
20. Steiniger, A., Uhrmacher, A.M.: Composing variable structure models - A revision of COMO. In Ören, T.I., Kacprzyk, J., Leifsson, L.Ð., Obaidat, M.S., Koziel, S., eds.: *SIMULTECH*, SciTePress (2013)
21. OSIsoft: The power of Asset Framework. Technical report, OSIsoft, LLC (2015) Published at <https://resources.osisoft.com/white-papers/the-power-of-asset-framework/>.
22. Cameron, D.B., Waaler, A., Komulainen, T.M.: Oil and Gas digital twins after twenty years. How can they be made sustainable, maintainable and useful? In: *SIMS 59*, Linköping University Electronic Press (2018)
23. Heaton, J., Parlikad, A.K.: Asset information model to support the adoption of a digital twin: West cambridge case study. *IFAC-PapersOnLine* **53**(3) (2020)
24. Rotondi, M., Cominelli, A., Di Giorgio, C., Rossi, R., Vignati, E., Carati, B.: The benefits of integrated asset modelling: lessons learned from field cases. In: *Europec/EAGE Conf. and Exhibition, OnePetro* (2008)
25. Wei, K., Sun, J.Z., Liu, R.J.: A review of asset administration shell. In: *IEEM*. (2019)
26. Fjøsna, E., Waaler, A.: READI Information modelling framework (IMF). Asset Information Modelling Framework. Technical report, READI Project (2021)
27. Mehmandarov, R., Waaler, A., Cameron, D., Fjellheim, R., Pettersen, T.B.: A semantic approach to identifier management in engineering systems. In: *Big Data, IEEE* (2021)
28. Poggi, A., Lembo, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: Linking data to ontologies. *J. Data Semant.* **10** (2008)
29. Skjæveland, M.G., Giese, M., Hovland, D., Lian, E.H., Waaler, A.: Engineering ontology-based access to real-world data sources. *J. Web Semant.* **33** (2015)
30. Wiedau, M., von Wedel, L., Temmen, H., Welke, R., Papakonstantinou, N.: EN-PRO Data Integration: Extending DEXPI Towards the Asset Lifecycle. *Chemie Ingenieur Technik* **91**(3) (2019)
31. IOGP JIP 36: CFIHOS Standards. <https://www.jip36-cfihos.org/cfihos-standards/> Accessed: 2021-12-12.
32. Bickford, J., Van Bossuyt, D.L., Beery, P., Pollman, A.: Operationalizing digital twins through model-based systems engineering methods. *Systems Engineering* **23**(6) (2020)
33. Nigischer, C., Bougain, S., Riegler, R., Stanek, H.P., Grafinger, M.: Multi-domain simulation utilizing SysML: state of the art and future perspectives. *Procedia CIRP* **100** (2021)
34. Delgoshaei, P., Austin, M.A., Veronica, D.A.: A Semantic Platform Infrastructure for Requirements Traceability and System Assessment. In: *ICONS, IARIA* (2014)
35. Fraga, A., Llorens, J., Alonso, L., Fuentes, J.M.: Ontology-Assisted Systems Engineering Process with Focus in the Requirements Engineering Process. In: *Complex Systems Design & Management, Springer* (2015)

# Are Formal Contracts a useful Digital Twin of Software Systems?\*

Jonas Schiffl<sup>[0000–0002–9882–8177]</sup> and Alexander Weigl<sup>[0000–0001–8446–4598]</sup>

Karlsruhe Institute of Technology, Karlsruhe, Germany  
{schiff1,weigl}@kit.edu

**Abstract.** Digital Twins are a trend topic in the industry today to either manage runtime information or forecast properties of devices and products. The techniques for Digital Twins are already employed in several disciplines of formal methods, in particular, formal verification, runtime verification and specification inference.

In this paper, we connect the Digital Twin concept and existing research areas in the field of formal methods. We sketch how digital twins for software-centric systems can be forged from existing formal methods.

**Keywords:** Formal Verification · Runtime Verification · Specification Mining · Temporal Logics

## 1 Introduction

**Motivation** *Digital Twin* is an emerging trend in many industries. The main focus is on the coupling of cyber-physical systems to a digital representation of the system, called the Digital Twin. The shape of a Digital Twin is tailored to the twinned cyber-physical system and the required reasoning. However, boundary constraints, such as performance restrictions, can also be involved. For example, a Digital Twin for predictive maintenance, i.e., the forecast of upcoming failures due to attrition, has different model elements from a Digital Twin for tracking the material flow.

Formal methods are a well-established research area. Although niche in industry adaption, they offer a rich toolbox for modeling systems. Therefore, formal methods are a natural candidate for digital twins of software systems. A recent survey paper states that only three of the surveyed approaches for Digital Twins for industrial automation systems use formal methods in contrast to 13 papers in the category “Exploratory investigation” and 17 in category “Testing” ([19, Table 1]).

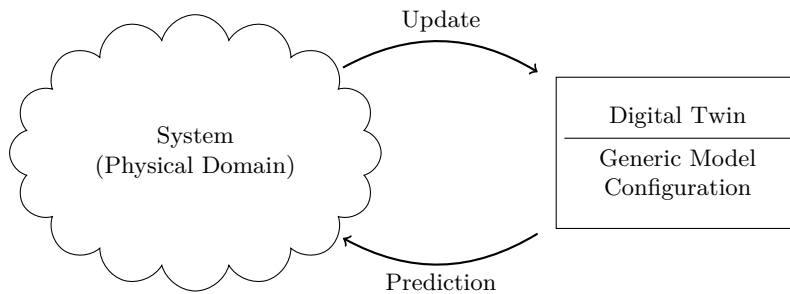
In this work, we show how formal methods, in particular formal contracts, can be used to build a Digital Twin of a software system, update it according to the state of the system, and derive predictions about system behavior and safety properties.

---

\* This work was supported by funding of the Helmholtz Association (HGF) through the Competence Center for Applied Security Technology (KASTEL) and by the research project SofDCar (19S21002), which is funded by the German Federal Ministry for Economic Affairs and Climate Action

**Contribution** In this paper, we apply several formal methods and tools to establish a framework for Digital Twins of reactive systems using formal software contracts. Therefore, we focus on twinning the software behavior of a reactive system operation according to its formal model, which we define through contracts.

The Digital Twin allows forecasting properties that depend on the behavior of the system. However, forecasting is not limited to the specific instance to which it is twinned. Moreover, knowledge learned from observation can be shared between Digital Twins, and What-if analyses are available. To achieve this, we show how existing formal methods, in particular formal verification, runtime verification, specification mining and program repair, can be combined for a Digital Twin.



**Fig. 1.** A Meta-model Digital Twin

**A meta-model for Digital Twin** For this paper, we assume a simple meta-model of Digital Twins, sketched in Figure 1. This meta-model consists of the original system that exists in the real world. Such systems can be cyber-physical systems, like cars, automated production systems, or energy systems. For the purposes of this discussion, we only consider pure software systems.

The Digital Twin is then a digital representation of an instance of a system. In case of a cyber-physical system, this can be a specific car or an individual transformer in the energy system. Coming from the domain of formal methods, we build the Digital Twin by using symbolic logic based models. We will later use contract-languages to represent the system. For software systems, the Digital Twin can be split into a generic part that is valid for all instances of the system parameters, and instance-specific configuration. For example, the generic consists of the classes and their contracts, but their use and composition is determined by the instance's configuration. Note that the Digital Twin also contains information about the environment. This is similar to the distinction between ABox and TBox in description logics. The TBox, the generic part, holds the axioms that are valid for all instances. The ABox, on the other hand, contains the information for the specific instance [9]. The TBox determines which parts of the ABox are relevant.



*Operations* The representation of a system as a Digital Twin should enable two operations:

- *Prediction* of selected properties. Prediction means reasoning whether a property holds for a specific instance or, due to the symbolic representation, for a family of instances. Moreover, we can perform “What-If” analyses: assuming a different configuration, we can predict properties on different instances, after change on the system. We can also model a transfer to a different environment. Prediction can lead to effects on the system, e.g., only configurations with a good predicted performance are deployed.
- The *Update* process consists of collecting and aggregating information from the (real-world) instance and transferring these into the configuration part of the model.

*Desired Properties* A Digital Twin also has to fulfill some properties. First, it should be faithful in the sense that the derived predictions are precise. If precision is too hard to achieve, we claim at least soundness: The prediction should be a pessimistic (or conservative) evaluation with respect to the safety analysis. The reasoning based on the Digital Twin should never attest safety falsely; if in doubt, it should rather predict non-existing safety issues.

Additionally, compositionality of a Digital Twin brings the same advantages as for the system. Systems can be built up by composition of multiple systems. The same is desirable for the Digital Twins: the Digital Twin of the top system is, ideally, a composition of the Digital Twins of the sub systems.

## 2 Contracts for Reactive Systems

In the following, we instantiate the Digital Twin meta-model for reactive software systems. For this, we define the individual components: the system and the formal model as well as the prediction and update processes.

**The system.** A reactive system is characterized by the following properties (cf. [10]): Their runtime is infinite or undetermined, and they interact with the physical world, e.g., by reading sensor values and controlling actuators. Typical representatives for reactive systems are embedded controllers in the automation or automotive domain.

Reactive systems can be constructed via composition from smaller systems. For example, Lingua Franca [18] defines reactive systems, named *reactors*, that can be built of other reactors, or an executable program fragment. A similar terminology can be found in IEC 61131-3 (the standard for programming languages for automated production systems) as *Function Block Diagrams*. In this paper, we assume the system model of [7], in which every system has an interface (input and output variables) and is composed of connected subsystems.

**The Digital Twin.** Cimatti et al. [7] provide OCRA, a tool that enables a design-by-contract methodology for reactive systems by defining components, their composition and associated contracts. OCRA focuses on the refinement relationship, thus verification of the program code against contract is left out.

Using the OCRA model, the Digital Twin of a system is its contract, which consists of two properties given in Linear Temporal Logic (LTL). The first property is an assumption on the system inputs. The second one is a guarantee of the system outputs under the given assumptions. Note that LTL is just the language we use for describing sets of traces. It can be replaced by any logic on traces (see below). The advantage of LTL is acceptance by a wide range of tools.

In addition to the contract, the Digital Twin is also aware of the structural architecture of the software given by the composition of subsystems and the information flow between them. Parameterization (or configuration) of a system is established via its input variables and by the selection of the sub-system during the composition.

The contract is a syntactical notion that describe the set of allowed behaviors of a system under a specified set of input traces. The behavior of a correct system is a subset of the allowed behavior. Often, due to the determinism in the system and the indeterminism (or abstraction) in the contract, the allowed behavior is indeed a strict superset of the actual actual behavior. Additionally, a system composition gives a refinement obligation, in which the composition, i.e., the collective of sub-systems, needs to conjointly fulfill the super system's contracts. Furthermore, the systems within a composition also need to be compatible with each other. Due to the over-approximating character of contracts, the Digital Twin tends to be imprecise, but sound for safety analyses. However, it is not possible to predict liveness properties. Note that proving the correctness of the system is sometimes infeasible, e.g., due to unavailability of the source code, complexity of the system (floating point, state explosion) or lack of knowledge of the environment. Additionally, we do not assume that the system is always compliant with the contract, especially, when the contracts are automatically mined from observation, or extracted from natural language and not carefully designed by the system creator and operators. This increases the need for a robust update that incrementally tighten the coupling between the instance of the system and Digital Twin built from contracts.

**Update** There are several update operations. The most simple one is detecting the parameterization of the instance, i.e., determining the values of the different inputs.

Moreover, the assumptions and guarantees in the contracts at the top-level and sub-systems needs to be validated during operation to ensure that the system operates in its expected boundaries. In particular, we check whether a system is correctly invoked by its outer context or environment, and whether it behaves as specified. To achieve this, Runtime Verification, such as [2] for LTL, can be used. However, from Runtime Verification we can only learn about violation of or adherence to a property.

Normally, we want to establish the correctness of a system statically, but some properties are too hard to be established in advance. In the best case, unexpected specification-violating runtime traces are recorded to enable specification mining. Specification mining is the discipline to extract LTL properties from traces. For example, Tuxedo [17] is a pattern-based approach, that instantiates LTL patterns with state formulas and validates them against the traces. An instantiated pattern must be fulfilled by a specified ratio of the observed traces to be considered valid. The valid instantiated patterns help to adjust the contracts. For example, in the case of a contract violation, we might consider weakening the assumptions of a system. On the other hand, specification mining may also be applied where the contracts are fulfilled. In this case, it can lead to a tightening of the contracts for a specific configuration or environment. Besides of mining specifications from recorded traces, there are tools for extracting LTL properties from natural languages, e.g. requirement documentation or user commands [22].

**Prediction** The prediction operation of our Digital Twin can be reduced to the well-established model-checking problem which allows us to verify the validity of a (LTL) property in a given Kripke structure. This covers the validation that each contract is adhered to by the associated system, and that systems are composed in a valid fashion.

But we benefit from the information learned during the Update operation: We can testify the contract adherence specifically in the used system configuration, and thus, we can save verification run-time without suffering a loss of verification validity for a specific instance. Furthermore, it may be beneficial to limit the verification process only to the recorded and mined environment of the instance to be validated.

Of course, the Digital Twin allows to conduct “What-If” analyses by altering the parameter of a system, or implementing it into a different environment. The latter simply requires changing the mined properties. This also requires tracking the origin of the formulas within the contracts.

But we are not limited to model-checking. Testing or simulation are also in reach by using the contracts. In particular, formal contracts enable testing and simulation even for software which is not executable in a simulation environment (e.g., due to inaccessible resources). There are also more sophisticated techniques. For example, *Program Repair* considers altering programs such that the program fulfills its contract. For this, the source code of the program is mutated until a suitable candidate is found [6,20]. A similar discipline is *Parameter Synthesis* [5].

**Limitations** By using assume-guarantee contracts and LTL as the specification language, we focus on describing behavior on a certain abstraction level. This allows us to reason well on the possibility to reach bad system states, but other properties are not predictable. For example, security properties that are

expressible as reachability are covered, i.e., integrity of the system, but confidentiality (e.g., secure information flow), and availability (liveness properties) are not. Moreover, runtime predictions, e.g., worst case execution time (WCET), are also not in our setup, due to the abstraction of time in LTL.

**Other contract languages.** Note that OCRA uses LTL, but its approach is not limited to a particular contract grammar. There are many variants of temporal logics. For example, Metric Temporal Logic (MTL) extends LTL to include real-time capabilities [14]. In MTL, temporal operators have an additional time span in which the formula needs to be fulfilled. MTL formulas are evaluated on “data words”—event traces with an explicit time value. MTL is well studied for runtime verification [21,11] and specification mining [12]. Additionally, Runtime Verification for MTL can also be quantitative by measuring how large the violation of the specification is [8].

Temporal logic formulas tend to be very hard to understand. Therefore, an additional goal is comprehensibility of the contracts and thus the Digital Twin. For example, Generalized Test Tables (GTTs) [4] are an engineer-friendly specification language derived from test tables used in the automation industry. A GTT describes a behavior in a particular scenario without the aspiration to be a complete specification. The rows in a GTT are the sequential steps of a test protocol, where each step consists of multiple assumptions and assertions (the table columns) for each input-output variable. Due to the table-based format, a fine-grained runtime monitoring is possible [24] in which the violation of single constraints can be tracked. Specification mining is currently not available, but might be adopted from approaches for learning finite-state machine based specifications (e.g., [15]).

### 3 Closing Remarks

In this paper, we present the idea of using formal contracts – existing in current design-by-contract approaches – for the representation of Digital Twins. Due to tool support, we focus on reactive systems and LTL, but the principle is also applicable to batch systems. For example, the Java Modeling Language (JML) [16] is an established specification language for Java source code with support of deductive verification [1] and runtime verification [13]. Specification mining (of method contracts) based on runtime information is currently not well-researched.

*Software and its refinement.* When we fade out the physical environment and concentrate on the software, we can state that the software, as it is digital, is itself the most precise definition of its behavior, and can also be seen as its own contract. Every other contract for a system over-approximates. But the abstraction of contracts is needed, as it allows us to hide the complexity of the implementation. Indeed, more abstract software models are often used in software engineering: For example, the buildability of a user-configurable product

is defined by a feature model. Feature models reflect the compatibility of software modules, which also depends on the behavior of the software.

*Evolution of Digital Twins with Relational Verification.* For our framework, we have only considered the classical specification and verification of single traces. A possible extension is the verification of relational properties [3] (or multi-properties). A relational property describes the relationship between multiple runs of the same or different systems. For example, regression verification – a relaxed program equivalence – is a relational property which helps to identify the introduction of unintended behavior during the co-evolution of hard- and software. Regression verification requires a description of the relationship between the common behavior of both systems [23]. As the regression contracts help with the co-evolution by coupling the old to the new version, they also support the evolution step of the Digital Twins by coupling mined knowledge from the old twin to the new twin.

## References




1. Ahrendt, W., Beckert, B., Bubel, R., Hähnle, R., Schmitt, P.H., Ulbrich, M. (eds.): *Deductive Software Verification - The KeY Book - From Theory to Practice*, Lecture Notes in Computer Science, vol. 10001. Springer (2016). <https://doi.org/10.1007/978-3-319-49812-6>
2. Bauer, A., Leucker, M., Schallhart, C.: Runtime verification for LTL and TLTL. *ACM Trans. Softw. Eng. Methodol.* **20**(4) (sep 2011). <https://doi.org/10.1145/2000799.2000800>
3. Beckert, B., Ulbrich, M.: Trends in relational program verification. In: Müller, P., Schaefer, I. (eds.) *Principled Software Development - Essays Dedicated to Arnd Poetzsch-Heffter on the Occasion of his 60th Birthday*. pp. 41–58. Springer (2018). [https://doi.org/10.1007/978-3-319-98047-8\\_3](https://doi.org/10.1007/978-3-319-98047-8_3)
4. Beckert, B., Ulbrich, M., Vogel-Heuser, B., Weigl, A.: Generalized test tables: A domain-specific specification language for automated production systems. In: Seidl, H., Liu, Z., Pasareanu, C.S. (eds.) *Theoretical Aspects of Computing - ICTAC 2022 - 19th International Colloquium, Tbilisi, Georgia, September 27-29, 2022, Proceedings*. Lecture Notes in Computer Science, vol. 13572, pp. 7–13. Springer (2022). [https://doi.org/10.1007/978-3-031-17715-6\\_2](https://doi.org/10.1007/978-3-031-17715-6_2)
5. Bezdek, P., Benes, N., Barnat, J., Cerná, I.: LTL parameter synthesis of parametric timed automata. In: Nicola, R.D., Kühn, E. (eds.) *Software Engineering and Formal Methods - 14th International Conference, SEFM 2016, Held as Part of STAF 2016, Vienna, Austria, July 4-8, 2016, Proceedings*. Lecture Notes in Computer Science, vol. 9763, pp. 172–187. Springer (2016). [https://doi.org/10.1007/978-3-319-41591-8\\_12](https://doi.org/10.1007/978-3-319-41591-8_12)
6. Brizzio, M., Degiovanni, R., Cordy, M., Papadakis, M., Aguirre, N.: Automated repair of unrealisable LTL specifications guided by model counting. *CoRR* **abs/2105.12595** (2021)
7. Cimatti, A., Dorigatti, M., Tonetta, S.: OCRA: A tool for checking the refinement of temporal contracts. In: Denney, E., Bultan, T., Zeller, A. (eds.) *2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013, Silicon Valley, CA, USA, November 11-15, 2013*. pp. 702–705. IEEE (2013). <https://doi.org/10.1109/ASE.2013.6693137>

8. Dokhanchi, A., Hoxha, B., Fainekos, G.: On-line monitoring for temporal logic robustness. In: Bonakdarpour, B., Smolka, S.A. (eds.) Runtime Verification - 5th International Conference, RV 2014, Toronto, ON, Canada, September 22-25, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8734, pp. 231–246. Springer (2014). [https://doi.org/10.1007/978-3-319-11164-3\\_19](https://doi.org/10.1007/978-3-319-11164-3_19)
9. Giacomo, G.D., Lenzerini, M.: TBox and ABox reasoning in expressive description logics. In: Padgham, L., Franconi, E., Gehrke, M., McGuinness, D.L., Patel-Schneider, P.F. (eds.) Proceedings of the 1996 International Workshop on Description Logics, November 2-4, 1996, Cambridge, MA, USA. AAAI Technical Report, vol. WS-96-05, pp. 37–48. AAAI Press (1996)
10. Halbwachs, N.: Synchronous programming of reactive systems. In: Hu, A.J., Vardi, M.Y. (eds.) Computer Aided Verification, 10th International Conference, CAV '98, Vancouver, BC, Canada, June 28 - July 2, 1998, Proceedings. Lecture Notes in Computer Science, vol. 1427, pp. 1–16. Springer (1998). <https://doi.org/10.1007/BFb0028726>
11. Ho, H., Ouaknine, J., Worrell, J.: Online monitoring of metric temporal logic. In: Bonakdarpour, B., Smolka, S.A. (eds.) Runtime Verification - 5th International Conference, RV 2014, Toronto, ON, Canada, September 22-25, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8734, pp. 178–192. Springer (2014). [https://doi.org/10.1007/978-3-319-11164-3\\_15](https://doi.org/10.1007/978-3-319-11164-3_15)
12. Hoxha, B., Dokhanchi, A., Fainekos, G.: Mining parametric temporal logic properties in model-based design for cyber-physical systems. *Int. J. Softw. Tools Technol. Transf.* **20**(1), 79–93 (2018). <https://doi.org/10.1007/s10009-017-0447-4>
13. Hussain, F., Leavens, G.T.: temporaljmlc: A JML runtime assertion checker extension for specification and checking of temporal properties. In: Fiadeiro, J.L., Gnesi, S., Maggiolo-Schettini, A. (eds.) 8th IEEE International Conference on Software Engineering and Formal Methods, SEFM 2010, Pisa, Italy, 13-18 September 2010. pp. 63–72. IEEE Computer Society (2010). <https://doi.org/10.1109/SEFM.2010.15>
14. Koymans, R.: Specifying real-time properties with metric temporal logic. *Real Time Syst.* **2**(4), 255–299 (1990). <https://doi.org/10.1007/BF01995674>
15. Le, T.B., Lo, D.: Deep specification mining. In: Tip, F., Bodden, E. (eds.) Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2018, Amsterdam, The Netherlands, July 16-21, 2018. pp. 106–117. ACM (2018). <https://doi.org/10.1145/3213846.3213876>
16. Leavens, G.T., Cheon, Y., Clifton, C., Ruby, C., Cok, D.R.: How the design of JML accommodates both runtime assertion checking and formal verification. *Sci. Comput. Program.* **55**(1-3), 185–208 (2005). <https://doi.org/10.1016/j.scico.2004.05.015>
17. Lemieux, C., Park, D., Beschastnikh, I.: General LTL specification mining (T). In: Cohen, M.B., Grunske, L., Whalen, M. (eds.) 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015. pp. 81–92. IEEE Computer Society (2015). <https://doi.org/10.1109/ASE.2015.71>
18. Lohstroh, M., Menard, C., Bateni, S., Lee, E.A.: Toward a Lingua Franca for deterministic concurrent systems. *ACM Trans. Embed. Comput. Syst.* **20**(4) (may 2021). <https://doi.org/10.1145/3448128>
19. Löcklin, A., Müller, M., Jung, T., Jazdi, N., White, D., Weyrich, M.: Digital twin for verification and validation of industrial automation systems – a survey. In: 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). vol. 1, pp. 851–858 (2020). <https://doi.org/10.1109/ETFA46521.2020.9212051>

20. Mironovich, V., Buzdalov, M., Vyatkin, V.: Automatic generation of function block applications using evolutionary algorithms: Initial explorations. In: 15th IEEE International Conference on Industrial Informatics, INDIN 2017, Emden, Germany, July 24-26, 2017. pp. 700–705. IEEE (2017). <https://doi.org/10.1109/INDIN.2017.8104858>
21. Thati, P., Rosu, G.: Monitoring algorithms for metric temporal logic specifications. In: Havelund, K., Rosu, G. (eds.) Proceedings of the Fourth Workshop on Runtime Verification, RV@ETAPS 2004, Barcelona, Spain, April 3, 2004. Electronic Notes in Theoretical Computer Science, vol. 113, pp. 145–162. Elsevier (2004). <https://doi.org/10.1016/j.entcs.2004.01.029>
22. Wang, C., Ross, C., Kuo, Y., Katz, B., Barbu, A.: Learning a natural-language to LTL executable semantic parser for grounded robotics. In: Kober, J., Ramos, F., Tomlin, C.J. (eds.) 4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA. Proceedings of Machine Learning Research, vol. 155, pp. 1706–1718. PMLR (2020)
23. Weigl, A., Ulbrich, M., Cha, S., Beckert, B., Vogel-Heuser, B.: Relational test tables: A practical specification language for evolution and security. In: Bae, K., Bianculli, D., Gnesi, S., Plat, N. (eds.) FormaliSE@ICSE 2020: 8th International Conference on Formal Methods in Software Engineering, Seoul, Republic of Korea, July 13, 2020. pp. 77–86. ACM (2020). <https://doi.org/10.1145/3372020.3391566>
24. Weigl, A., Ulbrich, M., Tyszberowicz, S.S., Klamroth, J.: Runtime verification of generalized test tables. In: Dutle, A., Moscato, M.M., Titolo, L., Muñoz, C.A., Perez, I. (eds.) NASA Formal Methods - 13th International Symposium, NFM 2021, Virtual Event, May 24-28, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12673, pp. 358–374. Springer (2021). [https://doi.org/10.1007/978-3-030-76384-8\\_22](https://doi.org/10.1007/978-3-030-76384-8_22)



# Digital Twin for Rescue Missions – a Case Study<sup>\*</sup>

Martin Leucker , Martin Sachenbacher , and Lars B. Vosteen 

Institute for Software Engineering and Programming Languages,  
Universität zu Lübeck, Lübeck, Germany  
{leucker,sachenbacher,vosteen}@isp.uni-luebeck.de

**Abstract.** In this paper, we explain through a case study how to develop a digital twin that can be used for safety analysis of missions in physical contexts. More specifically, we consider a scenario where firefighters are operating inside a building under fire, but communicating online with a mission control station. One of the main tasks of the mission control is to ensure that the firefighters always have enough oxygen to exit the building safely. To this end, a Digital Twin can be created that reflects the physical structure of the burning building, the location of the firefighters and the oxygen level in their breathing apparatus. The Digital Twin uses these models and a shortest path algorithm to estimate the oxygen required to exit the building, and alerts mission control and the respective firefighter to exit the building on time. The case study is used to illustrate key concepts for building Digital Twins in safety-critical contexts.

**Keywords:** Digital Twin · Safety Analysis · Complex Systems Engineering

## 1 Introduction

Digital Twins (DT) have gained significant attention in various industries, particularly those involving safety-critical operations. In this paper, we aim to demonstrate the practical application of a DT in monitoring a safety asset through a case study analysis. The scenario under consideration is firefighters communicating with a mission control team amid a burning building. A primary objective of the mission control team is to ensure that the firefighters have sufficient oxygen supply to evacuate the building safely. To facilitate this, we propose the implementation of a digital twin that reflects the physical characteristics of the building, the current location of the firefighters and the oxygen levels in their breathing apparatus.

We follow Feng et al. [9] and define a DT as a digital representation of a physical process that uses various techniques such as communication and data storage, data visualisation, modelling and calibration, state estimation, monitoring and what-if simulation to enhance the value of the physical system. The DT receives data from the physical system and maintains it for further analysis and

---

<sup>\*</sup> Funded by: German Federal Ministry for Economic Affairs and Climate Action, due to a resolution of the German Bundestag in the context of the project O5G-N-IoT

visualisation, presents the data clearly and concisely, uses mathematical models to simulate the behaviour of the physical system, estimates the current state of the system, continuously monitors the physical system and triggers alarms when predetermined conditions are met, and enables the user to simulate various scenarios and predict the expected outcome, thereby facilitating decision making.

This paper focuses on the problem class of safety-critical missions regarding a dedicated use case. Most of the different aspects of the use case have been considered in isolation. For example, [5,17,18,22] study indoor navigation [7] based on the building information model (BIM, [2]) or its fragment IFC [1]. The robot operating system (ROS) [24] has been extended to support navigation [11,16], also using the BIM model. Related to the problem of firefighter support is the indoor-emergency-navigation-system for complex buildings [23], which again uses BIM. Note that [6] gives an overview of state of the art in BIM and Fire Safety Engineering.

While DT have proven to be beneficiary [10,13,20], it is still a challenge to design and build up suitable DT. The goal of this paper is not to provide yet another solution for supporting firefighter scenarios but to identify key artefacts occurring in this and similar use cases. We identify their mathematical nature and discuss corresponding formal modelling and analysis techniques as well as supporting tools. We identify that in our use case, we have to deal with

- building models
- discrete mathematical objects and optimisation, and
- physical processes, typically modelled as differential equations.

We discuss formal representations from a computer science perspective and tools to be used for realising the case study. We implement our case study using state of the art tools (ROS and Python) to validate our findings.

In Section 2, a case study is provided, motivating a problem suitable for using a DT; it is analysed regarding its artefacts, and main categories of objects and processes are determined. Their formal models and supporting tools are discussed in Section 3.1 and their integration in Section 3.2. In Section 4, we have implemented our case study to gain first practical insights.

The research is part of the O5G-N-IoT project<sup>1</sup>, which aims to enhance security components with 5G technology.

## 2 The case study in detail

Let us describe our case study in detail to derive the main kinds of artefacts informally before we identify their mathematical nature.

---

<sup>1</sup> <https://o5g-n-iot.de/>

## 2.1 A typical scenario

The case study presented in this paper deals with firefighters' rescue from a burning building. We depict our scenario for buildings ranging from one to about five storeys and approximately 200 m<sup>2</sup>, ensuring that a single team of firefighters is sufficient to deal with the fire. We assume that a fire is detected and reported to a central fire station, which sends out an appropriate response team. The team is usually divided into two functional groups: mission command and emergency personnel. The mission commander operates from the mission control centre, part of one of the fire vehicles outside the building. The emergency personnel can be a group of up to 40 firefighters who carry out different tasks inside the building (see Fig. 1). The main tasks we have in mind are rescuing people, checking for people to be rescued and containing the fire. The firefighters operating in the building are supplied with oxygen by a self-contained breathing apparatus with a compressed air tank. The firefighters are in constant contact with the mission command centre via a permanent voice radio. Since this is the aim of our project, we also assume that 5G will be used for both voice radio and a data link to each firefighter. The data link is used, among other things, to receive vital signs data from each firefighter. We assume that their position and level of the compressed air tank are transmitted to the mission control centre.

The current state of the art is for the mission command to manually record the position of each firefighter and the corresponding air pressure levels regularly by radio and to record these values on a board in the mission control centre. As soon as a critical pressure value is detected for a firefighter, he or she is instructed to leave the building (again by radio). The incident commander can also draw up a plan to search the entire building for people. Last but not least, the incident commander continuously monitors the spread of the fire. The project aims to digitise and improve the first two tasks.

## 2.2 Analysis of the scenario

Let us revisit the scenario described above. We identify the following artefacts:

- There is a mission commander in direct contact with the firefighters.
- There is a constant data link sending vital data such as air pressure and position in the building.
- There is a plan of the building used for coordination and mission planning.
- There are physical processes taking place, such as the deflation of the air containers and the spread of the fire.
- There are optimisation problems, such as mission planning to search the whole building.
- There is a critical property that needs to be checked, i.e. the assessment of the remaining oxygen level in relation to the distance/time required to evacuate the building.

To support the tasks in question, we plan to model these (and similar) scenarios using a DT and apply simulation and optimisation methods based on the DT. To this end, we use the following simplifications:

- We assume that our 5G network connection provides a reliable voice and data link.
- We assume there is an existing reliable indoor location solution – see [21,27] for an overview of current methods.
- We assume that a suitable 3D plan of the building is available, although we discuss different options in the next section.
- We consider “simple” physical processes, such as the emptying of the air reservoirs, but leave “complicated” physical phenomena to later studies. In particular, we assume a static digital plan to represent the building initially but ignore any changes to the building, for example, due to fire.
- As a concrete task, we only consider the intelligent estimation of the remaining oxygen level with respect to the time needed to leave the building.

Clearly, these are highly simplifying assumptions. However, the current setup already shows important challenges and basic solutions.

### 2.3 The artefacts to be formalised

We identify the need for

- modelling the structure (e.g. floor plan) and semantics (e.g. accessible doors and stairs) of a building to represent locations of people and plan (feasible) routes
- solving discrete optimisation problems, such as finding the shortest path to an exit and travelling salesman to search all rooms (although the latter is not discussed in this paper)
- modelling physical processes, such as draining a compressed air supply

In the next section, we discuss different modelling and digital solution options for these artefacts, both mathematical and using standard formats and tools, and their interplay. In Section 4 we then show a concrete implementation together with a first evaluation.

## 3 Formalisation options for the artefacts of digital twins

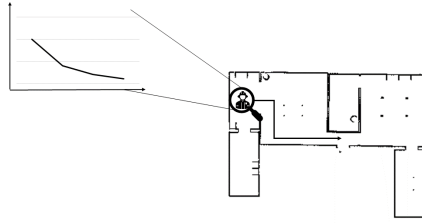
In the previous paragraph, we identified *building models*, *discrete optimisation problems*, and *physical processes* as the main artefacts to be formalised and addressed. Let us discuss the corresponding possibilities in the following.

### 3.1 The artefacts categories

*Building models* There are a wide range of standards and variants for the digital representation of buildings, from human-readable drawings to machine-usable 3D graphics with semantic metadata.

City Geography Markup Language (CityGML)<sup>2</sup> is an open standard that defines a conceptual model and exchange format for describing the geometry and

<sup>2</sup> <https://www.ogc.org/standards/citygml>



**Fig. 1.** Schematic representation of the mission, showing the three different types of artifact categories – building models, discrete optimisation processes (indoor navigation) and physical processes (loss of remaining air pressure)

appearance, topology (relationships, neighbourhoods) and semantics (meaning) of 3D city objects, facilitating the integration of urban geodata for applications in smart cities and digital city twins. It supports different levels of detail (LoD 0-3) so that objects become more detailed as the LOD increases to represent elements such as rooms, doors, corridors, stairs and even furniture. CityGML is based on standards from the Open Geospatial Consortium (OGC) and ISO 191XX.

Building Information Modelling (BIM) technology, as opposed to traditional CAD technology, can represent geometric and rich semantic information about building components and their relationships to support lifecycle data sharing. BIM is defined in ISO 29481-1:2016 as: “[the] use of a shared digital representation of a built object (including buildings, bridges, roads, process plants, etc.) to facilitate design, construction and operation processes to form a reliable basis for decisions.”

An important data exchange standard for BIM is the IFC (Industry Foundation Classes)<sup>3</sup> standard. The IFC object-based data model contains geometric and rich semantic information about building components and is supported by most BIM software vendors. A body of research has focused on extracting and managing semantic information about building components in the form of IFC for various applications, including indoor path planning [17].

The Green Building XML Schema (gbXML)<sup>4</sup> is an open schema developed to facilitate the transfer of building data stored in BIMs to engineering analysis tools. It is integrated into several computer-aided design (CAD) software packages, notably Autodesk. gbXML is a type of XML file with over 500 types of elements and attributes that can be used to describe all aspects of a building.

*Discrete optimisation processes* The optimisation problem described in the scenario is to find the shortest path to any existing exit. This can be seen as a

<sup>3</sup> <https://www.buildingsmart.org/>

<sup>4</sup> <https://www.gbxml.org/>

shortest path graph problem as the building model can be formed into a graph where each door is a node, and each direct connection between doors is a weighted edge. In the taxonomy provided in [19], the scenario is static, as the weights of the graph do not change over time.

*Physical processes* Physical processes are usually specified in terms of differential equations, which can then be solved explicitly in simple cases. In (real) more complex cases, this is usually not possible in an acceptable amount of time, so approximation algorithms are used.

This can be addressed in three types of approaches:

- Solving by hand or writing custom code adapted to the equation in question.
- Using dedicated libraries in appropriate programming languages (e.g. `scipy` [25] in Python, `dsolve` in Matlab or Mathematica).
- Use of languages specifically designed to describe physical processes (e.g. Modelica [12,4,8,3]). This can be complemented by the use of the Functional Mock-up Interface (FMI)<sup>5</sup> – a standard that defines a container and interfaces for exchanging dynamic simulation models from different modelling tools. It also specifies co-simulation Functional Mock-up Units (FMU), which contain the model and the simulation solver. In this way, simulation models with different time steps can be coupled.

In our setting, the digital twin needs to model physical processes to predict the depletion of compressed air over time, which may also depend on the type of activity, such as walking up or down stairs, whether additional equipment needs to be carried, and so on. Accurate modelling may require modular, multi-domain models of individual component models.

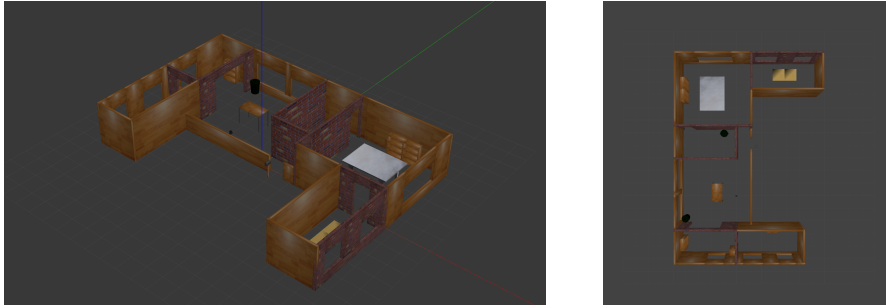
### 3.2 Integration of the three artefacts categories

In the previous subsection, we have identified key artefact categories of safety-critical rescue missions in buildings. However, it is essential that the concrete formalisation and supporting tools can be integrated into a single digital twin. A typical approach is encapsulating all artefacts as functional mock-up units and using a manually written integrator as coordinator. This approach has been mechanised by providing a programming layer to encapsulate simulators compliant with the FMI standard into OO structures, integrate FMOs into the class, and type systems [14].

## 4 Example implementation

To obtain practical insights into developing Digital Twins, we have implemented the discussed scenario following the discussion in the previous sections. To this extent, we focus on constantly checking whether each firefighter has enough air pressure to continue operating. To this end, two predictions must be made:

<sup>5</sup> <https://fmi-standard.org/>



**Fig. 2.** Design view of the building model used for experimentation

- How long will it take the person to reach the nearest exit?
- How long will the air last?

The localisation and the building model are used to calculate the time it will take to exit the building.

Therefore, based on the building plan, a graph can be created where passageways are nodes like doors, and all passageways directly connected by rooms are connected by edges weighted by distance. On this graph, a shortest path analysis can be performed from a given starting point to all possible exits, with the minimum result describing the shortest exit path. This can then be converted (e.g. by assuming a constant speed) into an estimated running time.

The pressure curve can be used to estimate how long the remaining air pressure in the tank will last. This is described by a monotonically decreasing function, which can be approximated, for example, by a polynomial.

The two durations must be determined periodically – in our scenario, with a buffer of a few minutes, low frequencies such as 0.1 Hz are sufficient since the emergency personnel cover a maximum of 10s equivalent distance within 10s and thus add a maximum of 20s time compared to immediate detection.

After subtracting the buffer, an alarm is triggered as soon as the time needed to evacuate the building falls below the time that breathing air is safely available.

Our implementation of the DT to support rescue missions uses the Robot Operating System (ROS)<sup>6</sup> as a way of determining routes in a building map and implements the air pressure forecast as a linear approximation. Currently, the implementation has the following limitations:

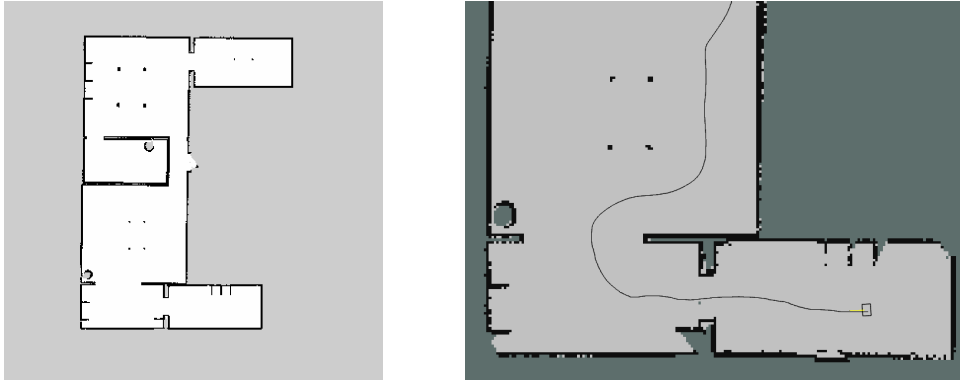
- Only one floor is taken into account.
- Building changes (e.g. caused by fire) are not considered, as the fire itself, and its ability to make pathways impassable are not considered.
- Air consumption is approximated as constant, ignoring influences such as load, fitness and environmental characteristics such as temperature.

A plan suitable for navigation is created based on a 3D model of a house (see Fig. 2). A file in IFC format can be reformatted [15] into a ROS-usable image file

<sup>6</sup> <https://www.ros.org/>



in PGM format using a toolkit called *ifcOpenShell*<sup>7</sup>. If a scale is known, it can be added to a ROS-usable YAML file with metadata. The resulting plan (see Fig. 3) is used as input for the ROS navigation stack<sup>8</sup>, and a path to a fixed location (like the exits) can then be calculated by calling the GetPlan service<sup>9</sup>. By calculating the length of the returned path and assuming a constant walking speed, the time required to leave the building via a selected exit can be estimated. In the case of multiple exits, the shortest path to an exit is used for further calculations.



**Fig. 3.** overview of the floor plan [26] of the building used in the simulation and a path found by ROS

A simple approximation of the remaining air pressure is obtained by assuming that the tank capacity is known and the respiration rate is constant; therefore, the remaining usable time of the gas container can be estimated. The elapsed time is then subtracted from this to obtain the remaining time.

In our prototype, both the time needed to leave the building and the time of remaining oxygen are calculated every ten seconds. A warning is issued if the difference falls below a safety margin of 300 seconds.

## 5 Discussion and outlook

In this paper, we have considered a real-world case study. We examined the artefact categories of Digital Twins and explored how to model them from a

<sup>7</sup> <https://ifcopenshell.org/>

<sup>8</sup> <https://wiki.ros.org/navigation>

<sup>9</sup> [https://docs.ros.org/en/api/nav\\_msgs/html/srv/GetPlan.html](https://docs.ros.org/en/api/nav_msgs/html/srv/GetPlan.html)

mathematical and computer science point of view. Additionally, we briefly outlined tool support and identified integration techniques. We do not claim any completeness of the overviews, yet hope to contribute a valuable contribution when building digital twins. We have implemented our scenario to gain practical insights and understand the limits of current approaches and tool support.

For our implementation, we considered several simplifications but also learned that many additional simplifications are needed to make the problem easier to handle. Notably, it was assumed that a suitable 3D model of the building already existed, that there was a stable radio link throughout the building, that sufficiently accurate indoor localisation was possible, and that the fire itself was disregarded entirely. While these simplifications have made the problem more tractable, it is important to note that, in reality, these assumptions may not hold (but may be overcome in future improvements/extensions of the approach). It is also important to note that no real-world experiments have yet been conducted, with only a simulation having been evaluated at this stage.

In future studies, conducting experiments in real-world environments would be beneficial to validate the proposed system. In addition, it seems useful to extend the Digital Twin to enable more comprehensive mission planning. The Digital Twin can simulate possible mission scenarios, assist in route planning for systematic searches, and estimate whether a task force can reach a specific location. This can lead to increased safety and efficiency in rescue operations.

Nevertheless, as our main finding, we learn that further tool support is essential to limit the burden of building up and employing digital twins.

## References

1. ISO 16739-1:2018, <https://www.iso.org/standard/70303.html>
2. ISO 19650-1:2018, <https://www.iso.org/standard/68078.html>
3. Blochwitz, T., Otter, M., Åkesson, J., Arnold, M., Clauss, C., Elmqvist, H., Friedrich, M., Junghanns, A., Mauss, J., Neumerkel, D., Olsson, H., Viel, A.: Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. Proceedings (09 2012). <https://doi.org/10.3384/ecp12076173>
4. Blochwitz, T., Otter, M., Arnold, M., Bausch, C., Clauß, C., Elmqvist, H., Junghanns, A., Mauss, J., Monteiro, M., Neidhold, T., Neumerkel, D., Olsson, H., Peetz, J.V., Wolf, S.: The functional mockup interface for tool independent exchange of simulation models. Proceedings of the 8th International Modelica Conference pp. 105–114 (03 2011). <https://doi.org/10.3384/ecp11063105>
5. Boysen, M., de Haas, C., Lu, H., Xie, X.: A journey from ifc files to indoor navigation. Web and Wireless Geographical Information Systems pp. 148–165 (2014)
6. Davidson, Anne, Gales, John: BIM and fire safety engineering - overview of state of the art. *International Journal of High-Rise Buildings* **10**(4), 251–263
7. El-Sheimy, N., Li, Y.: Indoor navigation: state of the art and future trends. *Satellite Navigation* **2**(1), 7. <https://doi.org/10.1186/s43020-021-00041-3>
8. Elsheikh, A., Awais, M.U., Widl, E., Palensky, P.: Modelica-enabled rapid prototyping of cyber-physical energy systems via the functional mockup interface. 2013 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems, MSCPES 2013 (05 2013). <https://doi.org/10.1109/MSCPES.2013.6623315>

9. Feng, H., Gomes, C., Thule, C., Lausdahl, K., Iosifidis, A., Larsen, P.G.: Introduction to digital twin engineering. 2021 Annual Modeling and Simulation Conference (ANNSIM) pp. 1–12. <https://doi.org/10.23919/ANNSIM52504.2021.9552135>
10. Gao, Y., Chang, D., Chen, C.H., Xu, Z.: Design of digital twin applications in automated storage yard scheduling. *Advanced Engineering Informatics* **51**, 101477 (2022). <https://doi.org/https://doi.org/10.1016/j.aei.2021.101477>, <https://www.sciencedirect.com/science/article/pii/S1474034621002275>
11. Gopee, M.A., Prieto, S.A., de Soto, B.G.: IFC-based generation of semantic obstacle maps for autonomous robotic systems, *Computing in Construction*, vol. 3. University of Turin, Rhodes, Greece (July 2022). <https://doi.org/10.35490/EC3.2022.161>, [https://ec-3.org/publications/conference/paper/?id=EC32022\\_161](https://ec-3.org/publications/conference/paper/?id=EC32022_161)
12. Jiang, H., Qin, S., Fu, J., Zhang, J., Ding, G.: How to model and implement connections between physical and virtual models for digital twin application. *Journal of Manufacturing Systems* **58** (06 2020). <https://doi.org/10.1016/j.jmsy.2020.05.012>
13. Jin, T., Sun, Z., Li, L., Zhang, Q., Zhu, M., Zhang, Z., Yuan, G., Chen, T., Tian, Y., Hou, X., Lee, C.: Triboelectric nanogenerator sensors for soft robotics aiming at digital twin applications. *Nature Communications* **11**(1), 5381. <https://doi.org/10.1038/s41467-020-19059-3>, <https://doi.org/10.1038/s41467-020-19059-3>
14. Kamburjan, E., Johnsen, E.B.: Knowledge structures over simulation units pp. 78–89 (2022). <https://doi.org/10.23919/ANNSIM55834.2022.9859490>, <https://doi.org/10.23919/ANNSIM55834.2022.9859490>
15. Kaulfuß, E.: Navigation of Mobile Robots for Interiorer Constructions Tasks. Diplomarbeit, Technische Universität Dresden (2021), page: 64
16. Kim, S., Peavy, M., Huang, P.C., Kim, K.: Development of bim-integrated construction robot task planning and simulation system. *Automation in Construction* **127**, 103720 (2021). <https://doi.org/https://doi.org/10.1016/j.autcon.2021.103720>, <https://www.sciencedirect.com/science/article/pii/S0926580521001710>
17. Lin, Y.H., Liu, Y.S., Gao, G., Han, X.G., Lai, C.Y., Gu, M.: The IFC-based path planning for 3D indoor spaces. *Advanced Engineering Informatics* **27**(2), 189–205 (2013)
18. Liu, L., Li, B., Zlatanova, S., van Oosterom, P.: Indoor navigation supported by the industry foundation classes (ifc): A survey. *Automation in Construction* **121**, 103436 (2021). <https://doi.org/https://doi.org/10.1016/j.autcon.2020.103436>, <https://www.sciencedirect.com/science/article/pii/S0926580520310165>
19. Madkour, A., Aref, W.G., ur Rehman, F., Rahman, M.A., Basalamah, S.M.: A survey of shortest-path algorithms. *ArXiv abs/1705.02044* (2017)
20. Madubuike, O.C., Anumba, C.J., Khallaf, R.: A review of digital twin applications in construction. *Journal of Information Technology in Construction* **27**, 145–172. <https://doi.org/10.36680/j.itcon.2022.008>, <https://www.itcon.org/paper/2022/8>
21. Obeidat, H., Shuaieb, W., Obeidat, O., Abd-Alhameed, R.: A review of indoor localization techniques and wireless technologies. *Wireless Personal Communications* **119**(1), 289–327. <https://doi.org/10.1007/s11277-021-08209-5>, <https://doi.org/10.1007/s11277-021-08209-5>
22. Palacz, W., Ślusarczyk, G., Strug, B., Grabska, E.: Indoor Robot Navigation Using Graph Models Based on BIM/IFC, pp. 654–665 (05 2019). [https://doi.org/10.1007/978-3-030-20915-5\\_58](https://doi.org/10.1007/978-3-030-20915-5_58)

23. Rueppel, U., Stuebbe, K.M.: BIM-based indoor-emergency-navigation-system for complex buildings. *Tsinghua Science and Technology* **13**, 362–367. [https://doi.org/10.1016/S1007-0214\(08\)70175-5](https://doi.org/10.1016/S1007-0214(08)70175-5), <http://ieeexplore.ieee.org/document/6073006/>
24. Stanford Artificial Intelligence Laboratory et al.: Robotic operating system, <https://www.ros.org>
25. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, , Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., Vijaykumar, A., Bardelli, A.P., Rothberg, A., Hilboll, A., Kloeckner, A., Scopatz, A., Lee, A., Rokem, A., Woods, C.N., Fulton, C., Masson, C., Häggström, C., Fitzgerald, C., Nicholson, D.A., Hagen, D.R., Pasechnik, D.V., Olivetti, E., Martin, E., Wieser, E., Silva, F., Lenders, F., Wilhelm, F., Young, G., Price, G.A., Ingold, G.L., Allen, G.E., Lee, G.R., Audren, H., Probst, I., Dietrich, J.P., Silterra, J., Webber, J.T., Slavič, J., Nothman, J., Buchner, J., Kulick, J., Schönberger, J.L., de Miranda Cardoso, J.V., Reimer, J., Harrington, J., Rodríguez, J.L.C., Nunez-Iglesias, J., Kuczynski, J., Tritz, K., Thoma, M., Newville, M., Kümmerer, M., Bolingbroke, M., Tartre, M., Pak, M., Smith, N.J., Nowaczyk, N., Shebanov, N., Pavlyk, O., Brodtkorb, P.A., Lee, P., McGibbon, R.T., Feldbauer, R., Lewis, S., Tygier, S., Sievert, S., Vigna, S., Peterson, S., More, S., Pudlik, T., Oshima, T., Pingel, T.J., Robitaille, T.P., Spura, T., Jones, T.R., Cera, T., Leslie, T., Zito, T., Krauss, T., Upadhyay, U., Halchenko, Y.O., Vázquez-Baeza, Y., SciPy 1.0 Contributors: SciPy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods* **17**(3), 261–272. <https://doi.org/10.1038/s41592-019-0686-2>, <https://doi.org/10.1038/s41592-019-0686-2>
26. Williams, C., Schroeder, A.: Utilizing ROS 1 and the turtlebot3 in a multi-robot system, <http://arxiv.org/abs/2011.10488>
27. Yassin, A., Nasser, Y., Awad, M., Al-Dubai, A., Liu, R., Yuen, C., Raulefs, R., Aboutanios, E.: Recent advances in indoor localization: A survey on theoretical approaches and applications. *IEEE Communications Surveys Tutorials* **19**(2), 1327–1346 (2017). <https://doi.org/10.1109/COMST.2016.2632427>

# A Digital Twin for Coupling Mobility and Energy Optimization: The ReNuBiL Living Lab<sup>\*</sup>

Daniel Thoma, Martin Sachenbacher, Martin Leucker, and Aliyu Tanko Ali

Institute for Software Engineering and Programming Languages,  
University of Lübeck, Lübeck, Germany  
{thoma,sachenbacher,leucker,aliyu.ali}@isp.uni-luebeck.de

**Abstract.** This paper presents a use case in the energy domain showing the benefits of digital twins. More specifically, we study the problem of peak shaving, which aims for managing a micro power grid in such a way that the energy demanded from the surrounding global power grid does not exceed certain limits. We examine a living lab consisting of university buildings as power consumers and power buffers in forms of fixed installed batteries as well as power-to-grid capable electrical vehicles that are booked by users. We provide a formal model of the relevant aspects of the micro grid and show how an artificial intelligence based prediction established from historical data as well as suitable simulation and optimization algorithms help to improve peak shaving.

**Keywords:** peak shaving · bi-directional charging · car sharing

## 1 Introduction

The ongoing transition to renewable and climate-neutral energy sources, such as wind and solar power, means that the production of electrical energy becomes more volatile and fluctuating on a daily and seasonal scale. This creates a need for technical solutions to intermediately store electrical energy, and to better forecast energy supply, in order to meet the demand.

However, technical solutions on the energy supply side alone will not be sufficient; instead, more flexibility on the consumption side and user participation will also be necessary. For instance in Germany, the largest electricity market in Europe, the installation of smart meters in private households will soon become mandatory and allow more consumers to shift load to times when energy is more abundant. For particularly energy-intensive devices, such as heat pumps or wallboxes for electric car charging, users get incentives if communication links allow grid providers to switch them off temporarily<sup>1</sup>.

Besides the energy sector, the mobility sector is still a large source of climate gas emissions. Battery-powered electric vehicles have the potential to reduce

---

<sup>\*</sup> This research was supported by the European Regional Development Fund (EFRE).

<sup>1</sup> Energy Industry Act (EnWG), §14a Network-oriented control of controllable consumption devices and controllable network connections, German Federal Ministry for Economic Affairs and Climate Action (BMWK), 2022

carbon emissions, if operated with green electricity; the positive effects can be even greater in a car sharing context when vehicles are shared among several users and thus the initial cost and carbon “backpack” of battery production is faster amortized. In addition, during idle times when the vehicles are parked and connected to the grid, their batteries can be used as buffers to store excess electricity and feed it back to the grid during peak demand times. Such so-called vehicle-to-grid concepts are now extensively studied in pilot projects [4, 7, 6] and corresponding norms to introduce bi-directional charging in the automotive market have recently been rolled out [2].

In this paper, we study a use case, on the campus of our university, that combines both the mobility and energy domain. The scenario consists of a fleet of electric vehicles with bi-directional charging capabilities and a stationary buffer battery, connected to a micro grid with additional consumers (buildings on the campus). The cars can be booked by users for trips, and the charging and discharging of the batteries (cars and stationary buffer) needs to be managed in such a way that the range of the cars suffices for the trips, while the total power demand of the micro grid should not exceed a given limit. The latter is called peak shaving and is important for grid stability, but also for electricity costs: grid usage fees, which make up a large proportion of the electricity costs, are based on the maximal power used in the billing period (monthly or annually), even if this maximum is reached only for a short period of time<sup>2</sup>.

It is easy in this scenario to devise a simple controller strategy that will, at each point in time, try to stay within the power limit by immediately reducing the charging power and – if this is not sufficient and there is energy left in the batteries – feeding back energy from the batteries into the grid. However, in our setting such a (myopic) controller might (depending on the additional load of other consumers in the micro grid) render user bookings infeasible by failing to charge the cars on time, or discharging them below the range required for upcoming bookings.

In the following, we present a formal model of this problem and propose a digital twin [1] solution that uses AI-based load forecasting, simulation and optimization to intelligently improve the balance between peak shaving and user’s mobility needs. In particular, the approach will simulate the effect of bookings on the micro grid to assess new bookings requested by users, while at the same time safeguarding already committed bookings such that the cars will have enough range for the planned trips. The system has been prototypically implemented and experiments have been conducted with carsharing users in a living lab on our university’s campus.

The rest of the paper is organized as follows: Section 2 presents the case study and our living lab in more detail, and introduces a mathematical optimization model to describe the problem formally. Section 3 describes our proposed digital twin solution on this model. Section 4 concludes with a discussion and directions for further work.

---

<sup>2</sup> This policy is typical for many energy providers, and also the case for our campus’ energy provider.

## 2 Case study: The ReNuBiL living lab

In the EU-funded research project ReNuBiL<sup>3</sup> (living lab for user-oriented bi-directional charging), an infrastructure for experimenting with vehicle-to-grid concepts in the context of battery-powered electric vehicles shared among different users was set up on University of Lübeck’s campus. It consists of (see also Figure 1):

- a Nissan LEAF passenger car with a battery capacity of 62kWh and approximate range of 385km
- a Nissan e-NV200 transporter with a battery capacity of 40kWh and approximate range of 275km
- two EVTEC<sup>4</sup> “coffee&charge” bi-directional charging stations with 20kW power output each
- an EVTEC “save&charge” stationary (second-life) battery with a capacity of 24kWh

The vehicles are connected to the stations using CHAdeMo plugs (direct current) for charging and discharging. The lab components come with an embedded software (EVTEC “barista”) that can be used to control the charging and discharging power of the batteries (in the cars and the container), provided that the vehicles are idle (i.e. not booked by customers for trips) and the charging levels of the batteries are sufficiently high. The lab infrastructure is set up next to the largest lecture hall (Audimax) on the campus, and so the components are connected to the local micro-grid of this building that is part of the overall campus’ power grid. Electricity meters were installed to record the energy flows (charging and discharging power, battery charge levels, etc.) in the lab and the adjacent Audimax building. Also, the vehicles themselves log data about their current position and energy consumption. The data is collected periodically since January 2021 and stored in a time-series database.

The vehicles can be booked by users via the project partner StattAuto<sup>5</sup>, who operates a fleet of more than 200 cars in the region and has included the ReNuBiL vehicles in its car sharing system so they can be booked by any of their customers.

StattAuto’s current solution for safeguarding bookings is to leave a gap of three hours between bookings, enough for the vehicles to fully recharge. Clearly, this is not optimal from the point of view of the carsharing operator but also in terms of peak shaving.

*Problem Description.* In our tackled setting, the cars are rented by customers and picked up at and brought back to the charging stations. While customers are free to charge the cars during a rental at third party stations, they are unlikely to do so except for very long trips. Although it is technically possible to utilize

<sup>3</sup> <http://www.renubil.de>

<sup>4</sup> <https://www.evtec.ch/>

<sup>5</sup> <https://www.stattauto-hl.de/>





**Fig. 1.** ReNuBiL living lab on University of Lübeck’s campus with two electric cars, two bi-directional charging stations, and stationary battery container

the full charging power of the stations at all times, as outlined above our aim is to stay below a power consumption limit (in our experiments of 45 kW) for the micro-grid including the charging stations, the stationary battery and the Audimax building, and avoid any peaks above this threshold.

Batteries can be charged from the grid while simultaneously other batteries are discharged into the grid. We therefore have a multilevel optimization problem: our highest priority is to enable bookings we have already confirmed to users (safeguarding bookings). To this end, users have to provide their requested range with each booking. Our second priority is to utilize both, the buffer as well as the car batteries in order to avoid exceeding the power consumption limit (peak shaving). Our third priority is to keep the cars available for short notice bookings, i.e. keep the cars charged as much as possible.

*Formal Model.* We first developed a formal model of our scenario as depicted in Fig. 2. The model describes the charging and discharging behaviours of the batteries involved. We treat both the car batteries as well as the stationary buffer battery equivalently as they differ only in their ability to be booked by users. Parameters and variables are indexed by the battery id  $i$ . For each battery, the model has the following parameters:  $\text{bookings}_i$  is the set of associated bookings comprising a start ( $t_1$ ) and end time ( $t_2$ ) and a required driving distance  $r$ .  $\text{batCharge}_i$  and  $\text{disCharge}_i$  assign a maximal charging or discharging power respectively to each level of charge. These parameters allow us to model the power restrictions of the batteries (see also [9, 5]).  $\text{efficiency}_i$  assigns an efficiency factor to a level of (dis-)charging power to model the power loss during (dis-)charging.

<p><i>Parameters</i></p> <p>bookings<sub><i>i</i></sub> ⊆ {(<i>t</i><sub>1</sub>, <i>t</i><sub>2</sub>, <i>r</i>) ∈ <i>R</i><sub>0</sub><sup>+</sup> × <i>R</i><sub>0</sub><sup>+</sup> × <i>R</i><sub>0</sub><sup>+</sup>   <i>t</i><sub>1</sub> &lt; <i>t</i><sub>2</sub>}</p> <p>batCharge<sub><i>i</i></sub> : <i>R</i><sub>0</sub><sup>+</sup> → <i>R</i></p> <p>batDischarge<sub><i>i</i></sub> : <i>R</i><sub>0</sub><sup>+</sup> → <i>R</i></p> <p>efficiency<sub><i>i</i></sub> : <i>R</i><sub>0</sub><sup>+</sup> → [0, 1]</p> <p>demand<sub><i>i</i></sub> : <i>R</i><sub>0</sub><sup>+</sup> → <i>R</i></p> <p><i>c</i><sub><i>i</i></sub> : <i>R</i><sub>0</sub><sup>+</sup></p> <p><i>Constraints</i></p> <p>(1) energy(0)<sub><i>i</i></sub> = <i>c</i><sub><i>i</i></sub></p> <p>(2) ∀(<i>t</i><sub>1</sub>, <i>t</i><sub>2</sub>, <i>r</i>) ∈ bookings<sub><i>i</i></sub> : energy<sub><i>i</i></sub>(<i>t</i><sub>2</sub>) = energy<sub><i>i</i></sub>(<i>t</i><sub>1</sub>) + demand<sub><i>i</i></sub>(<i>r</i>)</p> <p>(3) ∀(<i>t</i><sub>1</sub>, <i>t</i><sub>2</sub>, <i>r</i>) ∈ bookings<sub><i>i</i></sub>, <i>t</i><sub>1</sub> &lt; <i>t</i> &lt; <i>t</i><sub>2</sub> : energy<sub><i>i</i></sub>(<i>t</i>) = 0</p> <p>(4) ∀(<i>t</i><sub>1</sub>, <i>t</i><sub>2</sub>) ∈ between(bookings<sub><i>i</i></sub>), <i>t</i><sub>1</sub> &lt; <i>t</i> ≤ <i>t</i><sub>2</sub> : energy<sub><i>i</i></sub>(<i>t</i>) = energy<sub><i>i</i></sub>(<i>t</i><sub>1</sub>) + ∫<sub><i>t</i><sub>1</sub></sub><sup><i>t</i></sup> efficiency<sub><i>i</i></sub>(power<sub><i>i</i></sub>(<i>t</i>)) power<sub><i>i</i></sub>(<i>t</i>) dt</p> <p>(5) ∀(<i>t</i><sub>1</sub>, <i>t</i><sub>2</sub>) ∈ between(bookings<sub><i>i</i></sub>), <i>t</i><sub>1</sub> &lt; <i>t</i> ≤ <i>t</i><sub>2</sub> : batDischarge<sub><i>i</i></sub>(energy<sub><i>i</i></sub>(<i>t</i>)) ≤ power<sub><i>i</i></sub>(<i>t</i>) ≤ batCharge<sub><i>i</i></sub>(energy<sub><i>i</i></sub>(<i>t</i>))</p> <p>(6) ∀(<i>t</i><sub>1</sub>, <i>t</i><sub>2</sub>, <i>r</i>) ∈ bookings : charge<sub><i>i</i></sub>(<i>t</i><sub>1</sub>) ≥ demand(<i>r</i>)</p> <p>(7) power(<i>t</i>) = ∑<sub><i>i</i></sub> power<sub><i>i</i></sub>(<i>t</i>)</p>	<p><i>Variables</i></p> <p>energy<sub><i>i</i></sub> : <i>R</i><sub>0</sub><sup>+</sup> → <i>R</i><sub>0</sub><sup>+</sup></p> <p>power<sub><i>i</i></sub> : <i>R</i><sub>0</sub><sup>+</sup> → <i>R</i></p> <p><i>Goal</i></p> <p><i>f</i> : (<i>R</i> → <i>R</i>) → <i>R</i></p> <p><i>f</i>(<i>g</i>) = ∫<sub>0</sub><sup>∞</sup> max(0, (<i>g</i>(<i>t</i>) - limit(<i>t</i>))) dt</p> <p>minimize: <i>f</i>(power)</p>
---	---

**Fig. 2.** Formal model of the peak-shaving problem for bi-directional charging and car-sharing

demand<sub>*i*</sub> maps driving distance to energy demand and *c*<sub>*i*</sub> specifies the initial charge energy of the battery.

The variables of our models are two functions: the charge energy of a battery by time energy<sub>*i*</sub> and the current (dis-)charge power of a battery by time power<sub>*i*</sub>. The model only is defined in terms of the energy stored in the batteries not the actual charge as the energy stored by charge varies with voltage. Charging and discharging are distinguished by positive and negative power values.

The possible solutions for the power and energy functions are now defined by (1) the initial energy, (2) the consumption of required energy during bookings, (3) the inability to use car batteries during bookings, (4) the charging/discharging according to the assigned power with respective efficiency when batteries are available, (5) the power restrictions of the batteries and (6) the requirement to provide the required energy for bookings before they start.

Optimization goals can then be expressed as functions reducing the accumulative power function power defined by (7). The optimization goal to minimize the excess of a power limit can be expressed by the function *f* defined in Fig. 2. The limit there depends on time and therefore can take the external power consumption into account.

*Charge Strategy.* During operation of the charging station the (dis-)charging of the batteries has to be constantly optimized according to the charging model and our optimization goals. Solving the constraint model on the fly can be difficult to impossible especially considering the non-linear behavior of battery constraints. We therefore designed a dedicated charge strategy that achieves our goals comprising the following rules (stated here only informally due to lack of space):

1. for each battery, if we need to start charging at full power in order to facilitate the next (or a subsequent) booking, do so.
2. for each battery that remains, charge if we are below the limit, discharge, if we are above the limit.

3. prefer car batteries when charging, prefer buffer battery when discharging.

As our optimization goal is linear, i.e. exceeding the limit moderately for a long time is not better than exceeding the limit excessively for a short time, rules 1. and 2. result in an optimal strategy. Rule 3 deviates from that slightly to also optimize for availability for short notice bookings.

*Booking Assessment.* In contrast to other work such as [3], we are not considering a scheduling problem here: bookings are not scheduled but have to be assessed and facilitated when they are requested by the users. When a user wants to book a car, we provide him with a rating of that booking based on the optimization goal. This rating is computed by simulating the current bookings excluding and including the new booking according to the constraint model and the charge strategy. We then take the difference between the values of the goal function, i.e. the additional violation of the peak shaving power limit caused by adding the new booking. For high ratings, i.e. bookings that would force us to violate the limit by a large amount, we ask the user to consider changing his booking to a different time slot. The simulation incorporates a prediction of the future external power consumption, which is either done directly on multiple historical data traces or on a prediction generated by machine learning from these traces.

### 3 Proposed Solution: Digital Twin Approach

Our solution is based on the idea of a digital and a physical twin. The physical twin is constituted by the cars, the charging station and the university grid. The digital twin is constituted by the formal model of the charging station and car batteries, and the historical consumption data of the university grid and the machine learning model based on that data.

The physical charging station is controlled by the charging strategy designed above. The strategy provides control outputs to the battery control units and receives measurements from them as well as from the meters of the university grid. In addition, it receives minimal charge requirements for the car batteries that have to be observed in order to facilitate the currently confirmed bookings. These requirements are computed in the digital twin, i.e. the formal model using a backward simulation of the charging process with maximal possible charging power.

The second use case of the digital twin is the booking process: when requesting a booking, the user is provided with a rating. This rating is calculated by simulating the charging strategy over a prediction of the power consumption of the university grid. We support two prediction schemes: generating example traces from a machine learning model using LSTM networks (see [8] for according details) or directly using a set of historic traces. The rating presented to the user is then based on the average additional violation of the power consumption limit caused by the requested booking.

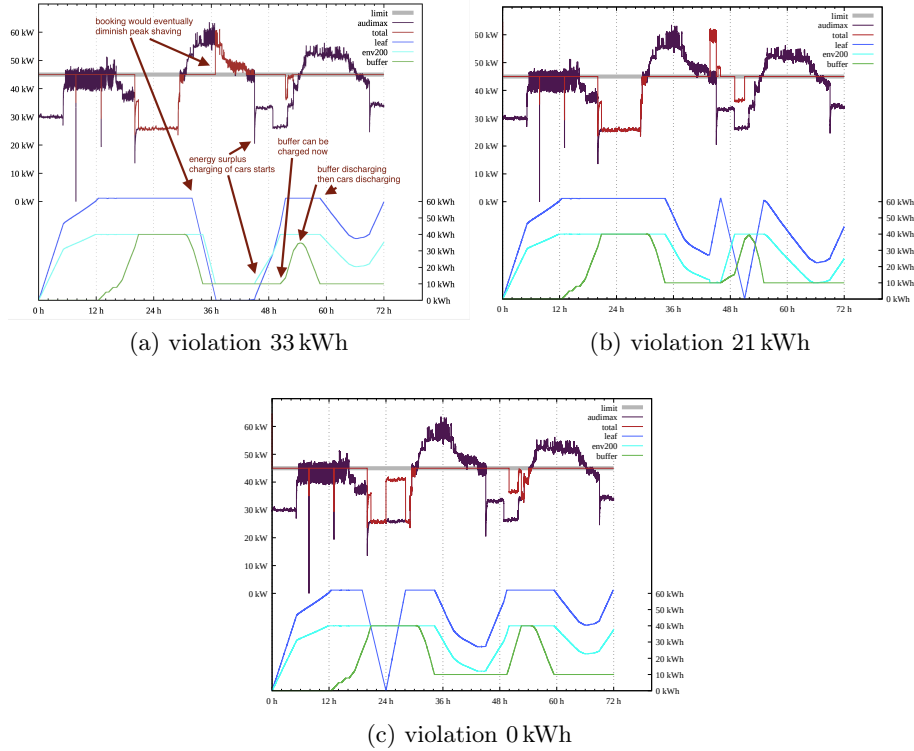


Fig. 3. Simulated scenario

### 3.1 Simulation and Experiments

We have simulated our solution extensively using the setup described above. Let us explain our approach using a typical simulation for three consecutive days Sunday 00:00h to Tuesday 24:00h of a typical week during the lecture period (week 2 of year 2023), as depicted in Fig. 3a.

Let us first concentrate on the upper part of the diagram. The purple curve shows the actual power consumption of the Audimax, our lecture hall acting as main consumer. We can see that the power demand is around 25-25 kW during night time and above 45 kW during day time with peaks up to 70 kW on work days. This curve serves as a prediction of the Audimax power consumption for the simulated scenario.

The red curve shows the simulated total power consumption that would arise from running our charging strategy with one booking request for the LEAF with the given Audimax power consumption. The considered booking request starts at 32 h with a duration of 5 h and requires the complete battery capacity of 62 kWh.

The anticipated limit of energy consumption is set to 45 kW as depicted by the grey horizontal line. Whenever this limit is reached, the energy of the fixed installed batteries (buffer) and those of the electric vehicles may be used to reduce the energy consumption from the global grid. Whenever surplus power is available, the battery may be charged. We can observe this behavior starting from 45 h. The energy levels of the batteries are depicted in the lower part of the diagram, blue and cyan for the LEAF and the e-NV200, respectively, and green for the buffer. As we can see, the system starts to charge the vehicle batteries as soon as surplus power is available. As there is not enough power available to charge the buffer as well and vehicles are prioritized, charging the buffer is delayed. Conversely, when the limit is reached at 54 h the buffer is discharged first.

Due to the booking, the LEAF is not available from 32 h to 37 h and is completely discharged after the booking. As a consequence after 36 h the system is not able to maintain the power limit and the total power starts to coincide with the Audimax power consumption until 45 h. Note that the batteries have a discharge limit of 10 kWh to avoid deep discharge. The system respects that limit but the vehicle batteries can be discharged further when driving.

In this scenario the limit would be violated by 33 kWh which in this case is due to the single booking. Fig. 3b depicts how the scenario would change if we were to move the booking to 46 h. Here, we can observe how the system prioritizes bookings over the power limit. The system manages to uphold the limit up to 46h, but due to the booking then has to switch to charging the battery of the LEAF resulting in a large violation. Consequently, the violation in this scenario is still 21 kWh.

Fig. 3c shows how moving the booking to 19 h, the previous evening, allows the vehicle to be charged over night, makes it available for power management during the day and avoids overshooting the power limit completely. In total, we see that intelligent peak shaving works out in many situations yet going beyond the limit could not be completely avoided by our system.

## 4 Discussion and Conclusion

Future energy systems will need a tighter and more intelligent integration between different sectors, in particular the sectors of mobility and electrical power. This includes comprehensive sensor meter gathering, data-driven trend analysis and forecasting, and real-time mathematical optimization of control parameters. Simultaneously, such systems must enable and support more flexibility on the user side, allowing consumers to express their desires/needs and receive relevant information that enables them to adjust their behavior accordingly, thus contributing to overall stability and sustainability goals.

In this paper, we presented an approach towards this goal in the setting of electric car sharing and bi-directional charging. The batteries of the cars can feed back their energy into the local micro grid, in order to limit the total power consumption (peak shaving). The users request to book the vehicles at certain

times and for certain desired ranges, such that they are not available as buffers for peak shaving during these times and also need to be re-charged, creating further load on the micro grid.

Our approach uses a digital twin model to balance the two conflicting concerns of optimal peak shaving and user mobility. The digital twin allows to simulate and assess user’s requests (at query time) and give recommendations to adapt their behavior (by possibly shifting their bookings to earlier or later times of the day). The user requests are typically issued several hours or days ahead of the actual bookings and so the evaluation/planning is based on predictions using historical data. In our setting, the objective to fulfill user’s bookings is prioritized over the objective of peak shaving. Thus, during execution time, the digital twin model is used to appropriately control charging and discharging (full-power charging to enable committed bookings, vs. reduced-power charging and discharging to do peak shaving).

As another partner in the ReNuBiL project, the Institute for Engineering Psychology<sup>6</sup> studies possible incentives to motivate users to adjust bookings and participate in peak shaving. Clearly the system could also be further optimized if users would be asked (and convinced) to re-schedule older, already committed bookings that turn out not to fit well with newer bookings. Due to the involved user interaction via the car sharing provider, this has not been considered so far.

Our current work includes the implementation of an alternative approach that uses constraint optimization on the formal model to synthesize optimal strategies, instead of selecting between pre-defined charge strategies (peak shaving and full-power charging). However, while this allows more flexibility and accuracy, the computational cost is much higher. Furthermore, we are trying to extend the machine learning approach to forecast not only the grid load, but also bookings, which is inherently difficult as much fewer training data is available.

## References

1. Feng, H., Gomes, C., Thule, C., Lausdahl, K., Iosifidis, A., Larsen, P.G.: Introduction to digital twin engineering. In: 2021 Annual Modeling and Simulation Conference (ANNSIM). pp. 1–12. IEEE (2021)
2. ISO 15118-1:2019 Road vehicles – Vehicle to grid communication interface – Part 1: General information and use-case definition. Standard, International Organization for Standardization (2019)
3. Klein, P.S., Schiffer, M.: Electric vehicle charge scheduling with flexible service operations (2022). <https://doi.org/10.48550/ARXIV.2201.03972>, <https://arxiv.org/abs/2201.03972>
4. Mwasilu, F., Justo, J.J., Kim, E.K., Do, T.D., Jung, J.W.: Electric vehicles and smart grid interaction: A review on vehicle to grid and renewable energy sources integration. *Renewable and sustainable energy reviews* **34**, 501–516 (2014)
5. Plett, G.L.: *Battery Management Systems*, vol. 1. Artech House Power Engineering and Power Electronics (2015)

---

<sup>6</sup> <https://www.imis.uni-luebeck.de/de>

6. Tepe, B., Figgner, J., Englberger, S., Sauer, D.U., Jossen, A., Hesse, H.: Optimal pool composition of commercial electric vehicles in v2g fleet operation of various electricity markets. *Applied Energy* **308** (2022)
7. Van Krieking, G., De Cauwer, C., Sapountzoglou, N., Coosemans, T., Messagie, M.: Peak shaving and cost minimization using model predictive control for uni-and bi-directional charging of electric vehicles. *Energy Reports* **7**, 8760–8771 (2021)
8. Walther, C.: Machine Learning for Time Series Prediction of Energy Data. Master’s thesis, Institute for Software Engineering and Programming Languages of the University of Lübeck, Germany (Nov 2021)
9. Weydanz, W., Jossen, A.: *Moderne Akkumulatoren richtig einsetzen* (in German). Reichardt Verlag (January 2006)

# Mining Digital Twins of a VPN Server

Andrea Pferscher<sup>1</sup>[0000-0002-9446-9541], Benjamin Wunderling<sup>1</sup>, Bernhard K. Aichernig<sup>1</sup>[0000-0002-3484-5584], and Edi Muskardin<sup>1,2</sup>[0000-0001-8089-5024]

<sup>1</sup>Institute of Software Technology, Graz University of Technology, Graz, Austria

<sup>2</sup>TU Graz - SAL DES Lab, Silicon Austria Labs, Graz, Austria

{aichernig, andrea.pferscher}@ist.tugraz.at,  
benjamin.wunderling@gmail.com, edi.muskardin@silicon-austria.com

**Abstract.** Virtual private networks (VPNs) are widely used to create a secure communication mode between multiple parties over an insecure channel. A common use case for VPNs is secure access to company networks. Therefore, bugs in VPN software are often severe. The Internet Key Exchange protocol (IKE) is a protocol in the Internet Protocol Security (IPsec) protocol suite used in VPNs. There are two version of IKE, IPsec-IKEv1 and the newer IPsec-IKEv2, with IPsec-IKEv1 still widely used in practice. While IPsec-IKEv2 has been investigated in the context of automata learning, no such work exists for IPsec-IKEv1. This paper closes the gap for the IPsec-IKEv1 protocol and shows the steps taken to learn a digital twin of an IPsec server using automata learning. We present and contrast two learned models of an IPsec server. Using learning, we also found security issues in encryption libraries.

**Keywords:** IPsec · VPN · Active automata learning · Digital twin · Model mining.

## 1 Introduction

Virtual Private Networks (VPNs) allow secure communication over an insecure channel. The importance of VPNs has increased dramatically since the COVID-19 pandemic due to the influx of people working from home [4]. This makes finding vulnerabilities in VPN software more critical than ever. In practice, VPNs are set up based on third-party components, which are usually closed source. In addition, multiple parties are involved in VPN communications. These challenges make it difficult to test possible security vulnerabilities in all scenarios.

Behavioral models are a useful tool for testing and verifying complex systems. A model can be viewed as a digital twin that simulates the behavior of the system. The availability of models, however, might be limited for the following reasons. First, the manual creation of a model can be a tedious and error-prone process. Second, the model must always be kept up to date.

Automata learning has proven itself as a useful technique for automatically generating behavioral models of various communication protocols, e.g., Bluetooth Low Energy [15], TLS [19], SSL [7], or MQTT [23]. Active learning techniques create a behavioral model by actively querying the system to gain knowledge about it. In this way, not only a model is created, but also a stateful testing



approach for black-box systems. The learned model represents a digital twin of the system under learning, which can then be used for other model-based techniques, like model-based test-case generation or model checking.

In this paper, we investigate the applicability of automata learning for mining security-critical components of a VPN. For this, we learn the behavioral model of the Internet Key Exchange protocol (IKE) protocol which is part of the Internet Protocol Security (IPsec) protocol suite. IKE is used to share authenticated key material between the involved parties. For learning, we use a VPN client to query the VPN server instantiation. This allows us to create a digital twin of the security-critical part of a VPN server without knowing its internals (black-box). The work is part of LearnTwins, a research project on learning digital twins<sup>1</sup>.

The paper is organized as follows. Sect. 2 provides details about the used modeling-formalism, the learning algorithm, and the system-under-learning. Section 3 provides a comparison with related work. In Sect. 4, we present our learning setup for mining a digital twin of a VPN. In Sect. 5 we present our learned models and, finally, in Sect. 6 we draw our conclusions.

## 2 Preliminaries

### 2.1 Mealy Machines

Mealy machines are a modeling formalism for reactive systems such as communication protocols. They are finite-state machines in which each transition is labeled with an input and the corresponding output action. We consider Mealy machines to describe deterministic behavior. More formally, a Mealy machine is defined as a 6-tuple  $M = \{S, s_0, I, O, \delta, \lambda\}$ , where  $S$  is a finite set of states,  $s_0 \in S$  is the initial state,  $I$  is a finite set called input alphabet,  $O$  is a finite set called output alphabet,  $\delta$  is the state-transition function  $\delta: S \times I \rightarrow S$  that maps a state and an element of the input alphabet to another state in  $S$  and  $\lambda$  is the output function  $\lambda: S \times I \rightarrow O$  that maps a state-input pair to an output in  $O$ .

### 2.2 Automata Learning

Automata learning algorithms generate a model that describes the behavior of the system under learning (SUL). We differentiate between active and passive automata learning. Passive learning creates a behavioral model based on a given data set, e.g., log files, while active learning directly queries the SUL. For learning a digital twin, we prefer active learning since active algorithms can query unusual or rare scenarios, whereas passive learning depends heavily on the provided data.

Today, many active learning algorithms are based on concepts derived from Angluin’s  $L^*$  algorithm [1], which was designed to identify deterministic finite automata formalizing regular languages. In her seminal work, Angluin introduced the concept of the minimally adequate teacher (MAT), in which a learner queries a teacher about the SUL to iteratively generate a behavioral model. The teacher

<sup>1</sup> <https://learntwins.ist.tugraz.at>

answers membership and equivalence queries posed by the learner regarding the SUL. Membership queries are used to check whether a word is accepted by the SUL. The learner then updates their model of the SUL based on the answers to their queries. Equivalence queries are used to check if a learned model exactly matches the behavior of the SUL. If the teacher provides a counterexample that represents the behavioral difference between the learned model and SUL, the learner improves the model by asking more membership queries. The MAT concept has been extended to facilitate other model formalisms like Mealy machines [21], where membership queries are renamed to output queries. Output queries are used to retrieve outputs to the corresponding input sequences.

The MAT concept is also used by other active learning algorithms. Kearns and Vazirani [12] propose an active learning algorithm that uses an underlying tree-based data structure to construct a model. We refer to their learning algorithm as *KV*. *KV*'s tree-based data structure can reduce the number of required output queries compared to the table-based technique used in the  $L^*$  algorithm.

### 2.3 Internet Protocol Security

Internet Protocol Security (IPsec) is a VPN Layer 3 protocol suite used to securely communicate over an insecure channel. It includes three sub-protocols: Internet Key Exchange protocol (IKE), the Authentication Header (AH), and the Encapsulating Security Payload (ESP) protocol. IKE is mainly used for authentication, and the secure exchange and management of keys. IKE has two versions, IKEv1 and IKEv2, with IKEv2 being the newer and recommended version [2]. Following a successful IKE round, either AH or ESP is used to send packets securely between parties. While AH only ensures the integrity and authenticity of messages, ESP also ensures their confidentiality through encryption.

While AH only ensures the integrity and authenticity of messages, ESP also ensures their confidentiality through encryption.

Compared to other protocols, IPsec offers a high degree of customizability, allowing it to be fitted for many use cases. However, in a cryptographic evaluation of the protocol, Ferguson and Schneier [5] criticize the complexity arising from the high degree of customizability as the biggest weakness of IPsec. To address its main criticism, IPsec-IKEv2 was introduced in RFC 7296 [11] to replace IKEv1. Nevertheless, IPsec-IKEv1, RFC 2409 [10], is still in widespread use to this day, with the largest router producer in Germany, AVM, still only supporting IKEv1 in their routers [8]. We investigate IPsec-IKEv1 with ESP in this paper and focus on the IKE protocol.

The IKEv1 protocol works in two main phases, both relying on the Internet Security Association and Key Management Protocol (ISAKMP). A typical key

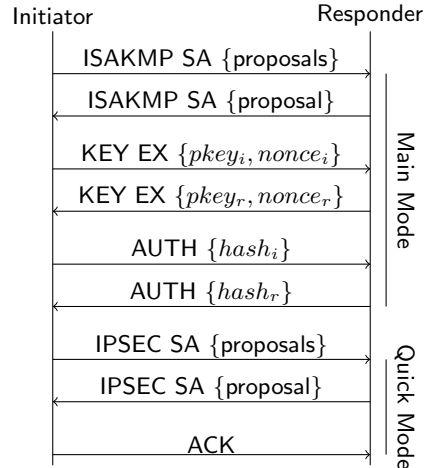


Fig. 1. IKEv1 between two parties

exchange between two parties, an initiator and a responder, can be seen in Fig. 1. In phase one (Main Mode), the initiator sends a Security Association (SA) to the responder. A SA essentially details important security attributes for a connection such as the encryption algorithm and key-size to use, as well as the authentication method and the used hashing algorithm.

These options are bundled in containers called proposals, with each proposal describing a possible security configuration. While the initiator can send multiple proposals to give the responder more options to choose from. In comparison, the responder must answer with only one proposal, provided it supports one of the suggestions. Subsequently, the two parties perform a Diffie-Hellman key exchange and exchange nonces to generate a shared secret key. This secret key is used as a seed key for all further session keys. Following a successful key exchange, all further messages are encrypted. Finally, both parties exchange hashed authentication material (usually pre-shared keys or certificates). If the hashes can be verified, a secure channel is created and used for phase two communication.

The shorter phase two (Quick Mode) begins with another SA exchange. This time, however, the SA describes the security parameters of the ensuing ESP/AH communication. This is followed by a single acknowledge message from the initiator to confirm the agreed upon proposal. After the acknowledgment, all further communication is done via ESP/AH packets.

### 3 Related Work

Model learning became a popular tool for creating behavioral models of various communication protocols, e.g., TLS [19], TCP [6], SSL [7], MQTT [23], 802.11 4-Way Handshake of Wi-Fi [22], or BLE [15]. The learned models reveal differences in the specification or provide a useful extension to the unspecified properties. In addition, the learned models serve as a basis for further techniques like model-checking or model-based security testing. In the VPN domain, Daniel et al. [3] learned a model of two OpenVPN implementations. The challenges of implementing a learning setup for OpenVPN were discussed by Novickis [14]. In contrast to our technique, they learned a more abstract model of the entire OpenVPN session, where details about the key exchange were abstracted in the learned model. Closely related to our work, Guo et al. [9] learned a model of the IPsec-IKEv2 protocol. The authors stress that IPsec-IKEv1 is more complicated to configure securely, which highlights the need to test the configuration of the older version as well, since it is still widely used [8]. With our work, we complete the learning approaches for all IKE versions.

### 4 Method

*Environment Setup.* As our SUL, we used a Linux Strongswan<sup>2</sup> US.9.5/K5.15.0-25-generic. Learning was done using two VirtualBox 6.1 virtual machines (VMs)

<sup>2</sup> <https://www.strongswan.org/>

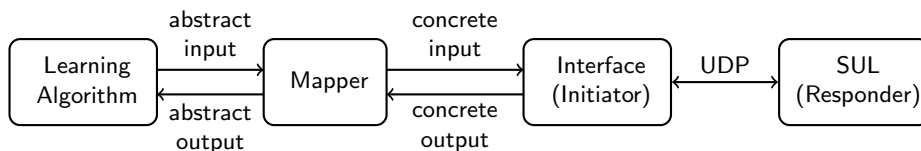


Fig. 2. Automata learning setup for learning a model of IPsec-IKEv1.

running standard Ubuntu 22.04 LTS distributions. Both VMs were allotted 4 GB of memory and one CPU core. All communication took place in an isolated virtual network to eliminate external influences. During learning, all power-saving options and similar potential causes of disruptions were disabled. The IPsec server was restarted before the start of each learning procedure to ensure identical conditions. We designated one VM as the initiator and one as the responder to create a typical client-server setup. The open-source IPsec implementation was installed on the responder VM and set to listen for incoming connections from the initiator VM. The Strongswan server was configured to use pre-shared keys for authentication and default recommended security settings. Additionally, it was set up to allow unencrypted notification messages, which we used to reset the connection during the learning process. For learning, we used the Python library AALPY [13] version 1.2.9 in conjunction with the packet manipulation library Scapy [20], version 2.4.5. Significant effort was invested into expanding the ISAKMP Scapy module to support all packets required for IPsec.

*Learning Setup.* Figure 2 gives an overview of the learning setup, adapted from Tappler et al. [23]. The learning algorithm sends abstract inputs chosen from the input alphabet to the mapper, which converts them to concrete inputs. The concrete inputs, which are actual IPsec packets, are then sent to the SUL, by means of a UDP communication interface. The interface represents the initiator whereas the SUL is the responder represented by a Strongswan server instance. This separation between abstract and concrete inputs and outputs allows us to learn a generic model in a reasonable amount of time.

The abstract inputs consist of the initiator-to-responder messages: `isakmp_sa`, `key_ex`, `auth`, `ipsec_sa` and `ack`. The responder-to-initiator outputs from Fig. 1 were extended by `NONE` and `ERROR`, where `NONE` signifies a lack of response from the SUL and `ERROR` is used as a collection of received error notifications. We use the KV [12] algorithm for learning with the improved counterexample-processing of Rivest and Schapire [18]. Since a perfect equivalence oracle cannot be assumed in practice, we substitute the equivalence oracle with model-based conformance testing between the intermediate learned model and the SUL. The conformance tests provide state-coverage combined with randomness.

Our mapper implements translation methods for each communication step in a typical IPsec-IKEv1 exchange, as described in Sect. 2. We use the Python library Scapy to construct IKEv1 packets. This approach allows us to change the fields and values of generated packets at will, opening up the possibility of fuzz testing these fields in future work as shown by Pferscher and Aichernig [16].

Parsing was made more difficult by the fact that Scapy does not support all the packets required by IPsec-IKEv1. To solve this problem, we implemented the missing packets in the Scapy ISAKMP class and used this modified version.

The IPsec packets generated by the mapper are then passed on to our interface for the SUL that handles all incoming and outgoing UDP packets. Additionally, it converts responses from the SUL into valid Scapy packets and passes them on to the mapper. The mapper class then parses the responses received from the interface and returns an abstract output representing the received data to the learning algorithm. Since part of communication is encrypted, we require a framework that correctly handles the en/decryption of messages. For each request, we store the base key and the responses for use in the next message, and update affected key material as needed. Most notably, the initialization vectors (IVs) are updated in almost every request and differ between messages. Informational requests also handle their IVs separately. For each request that we send, if available, we try to parse the response, decrypting it if necessary and resetting or adjusting internal variables as required to match the server.

Automata learning requires that the SUL can be reset to an initial state after each query. We implement this using a combination of the ISAKMP delete request and general ISAKMP informational error messages. While delete works for established connections in phase two of IKE, we require informational error messages to trigger a reset in phase one. Implementation was hindered at times by unclear RFC specifications, but this was overcome by manually comparing packet dumps and Strongswan logs to fix encryption errors.

*Combating Non-determinism.* During learning, the server occasionally exhibited non-deterministic behavior. For this, we implemented two methods of counteracting it. In our first method, we simply repeat the output query if we observe non-deterministic behavior. In this case, we select the output that occurs most often. Additionally, using timed waits after each input also helped to further decrease the number of non-determinism errors. However, learning still failed occasionally with these mitigation mechanisms for non-deterministic behavior.

A closer examination of the remaining non-deterministic behavior led to the discovery that it is caused by so-called retransmissions. Essentially, the IKE specification allows for previous messages to be retransmitted if deemed useful by the server. A possible trigger could be the final message of an IKE exchange being skipped/lost. For example, if instead of an AUTH message, the server receives a phase two IPSEC SA message, the server would not know if it missed a message or if there was an error on the other party's side. In this case, the Strongswan server reacts by retransmitting the previous message, prior to the missing one in an attempt to signal to the other party, that they should resend the missing message. To counteract this behavior, we implemented checks in our mapper to allow for the ignoring of retransmissions. If a repeated message ID is found, it is flagged as a retransmission. With this addition, the IPsec server behaved deterministically. The downside of this method is that it completely ignores the retransmissions, which could be a good source of information for fingerprinting different IPsec servers.

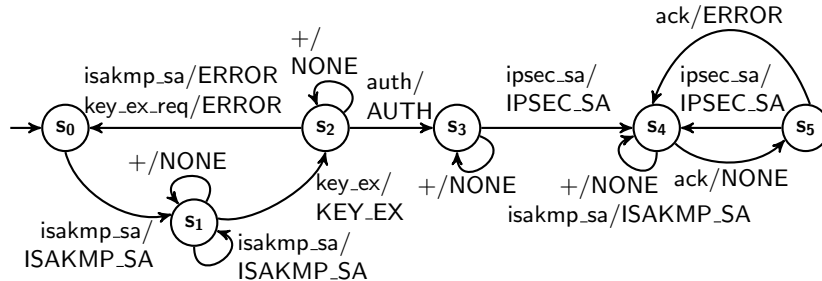
## 5 Evaluation

In our evaluation, we used two different learning setups to learn a model of the Linux Strongswan server. The first setup considers retransmitted messages from the server as outputs, whereas the second setup filters out any retransmitted messages. We included the model with retransmitted messages in the Appendix A, see Fig. 4. The model without retransmission is shown in Fig. 3. To enhance the readability of the models, we simplified the depicted models. The complete models are available as supplementary material [17].

For learning the models, we compared the active automata learning algorithms:  $L^*$  and KV. Both algorithms use an improved counterexample processing following Rivest and Schapire [18]. We prefer the usage of KV since it required fewer queries and less time to learn a model. Learning with KV was about one-fifth faster than  $L^*$  and required less than half the output queries. The model with retransmission (Fig. 4) has 13 states, whereas the model without retransmissions (Fig. 3) has only six states. For readability, we group together several different error messages in the output ERROR. The models also include NONE responses, which were used in cases where the input misses sensible information to establish an encrypted communication. While observing the behavior of the server when exposed to completely non-sensible input is interesting from a security testing standpoint, as all specifications state that the encryption requires a prior keying procedure, we decided to ignore those few cases. However, for future work in the field of fuzzing, these edge-cases should be considered as well.

We state the learning results of learning without retransmissions, since learning was more reliable in this case. We repeated each experiment five times and averaged the results. KV required approximately 38 minutes (2296 seconds) to learn, 79 output queries and 60 conformance tests were performed in 753 and 991 steps respectively. Learning performed four equivalences queries. The learning results for the other model can be found in the Appendix A.

Examining both models, especially Fig. 4, we can clearly see the separation between the two phases of the IKEv1 protocol. Phase one (Main Mode) completes in state  $s_3$ , and phase two (Quick Mode) begins right thereafter in state  $s_4$ . Comparing the models, we see that for the states  $s_0$  to  $s_3$  both models are identical. This is likely due to the fact, that most differences were caused by re-



**Fig. 3.** Simplified model of Strongswan server learned without retransmitted messages. The ‘+’-symbol is an abbreviation for all inputs that are not explicitly shown.

transmissions which only occur in phase two. The model depicted in Fig. 3 shows streamlined behavior that fits our reference IKE exchange (see Fig. 1) almost perfectly. The clean automaton makes it easy to see, that after a connection has been established, we can still create new connections or reestablish existing ones by sending another IPSEC SA message and then acknowledging the response.

The synthesis of a digital twin from the Linux Strongswan server not only provides interesting insight into the behavioral aspects of the implementation, especially regarding unexpected retransmitted messages. It also helped to test the general environment that is used to establish a VPN. A notable finding was the discovery of a very niche bug in a used Python Diffie-Hellman (DH) key exchange library<sup>3</sup>. The bug was very elusive and only found thanks to the exhaustive number of packets sent by the learning algorithm. It turns out there was a very niche bug in the library where, if the most significant byte (MSB) was a zero, it would be omitted from the response, causing the local result to be one byte shorter than the value calculated by the SUL. Regardless, it could compromise the security of affected systems and therefore the maintainer of the library has been notified of the problem. Due to the elusive nature of this bug, it would very likely not have been noticed without the exhaustive communication done by the model learning process and without seeing the resulting non-deterministic behavior of the SUL due to the truncated message.

## 6 Conclusion

We presented an automata learning framework that automates the generation of a digital twin of a VPN server. More precisely, the learned digital twin represents a behavioral model of the security-critical key-exchange procedure. Albeit creating a learning interface was not straightforward, the interface has to be created only once and can now be used to synthesize a new digital twin within less than one hour in case the VPN server is modified. Already the learning procedure of the digital twin revealed security issues in the libraries used to establish a secure VPN. The applications for these mined models are manifold. For example, the twin models can be used to simulate VPN components in a network infrastructure. This can be useful when we want to simulate differences in behavior between distinct VPN servers. Instead of setting up each VPN server individually, the mined models can be used. A second application could be model-based verification of the VPN server. By checking formal properties, gaps between the mined model and the VPN specification can be investigated. In future work, we want to use the digital twin to apply further model-based techniques, like stateful fuzz testing or model checking.

*Acknowledgement.* This work is supported by the LearnTwins project funded by FFG (Österreichische Forschungsförderungsgesellschaft) under grant 880852, and the “University SAL Labs” initiative of Silicon Austria Labs (SAL) and its Austrian partner universities for applied fundamental research for electronic based systems.

<sup>3</sup> <https://github.com/TOPDapp/py-diffie-hellman>

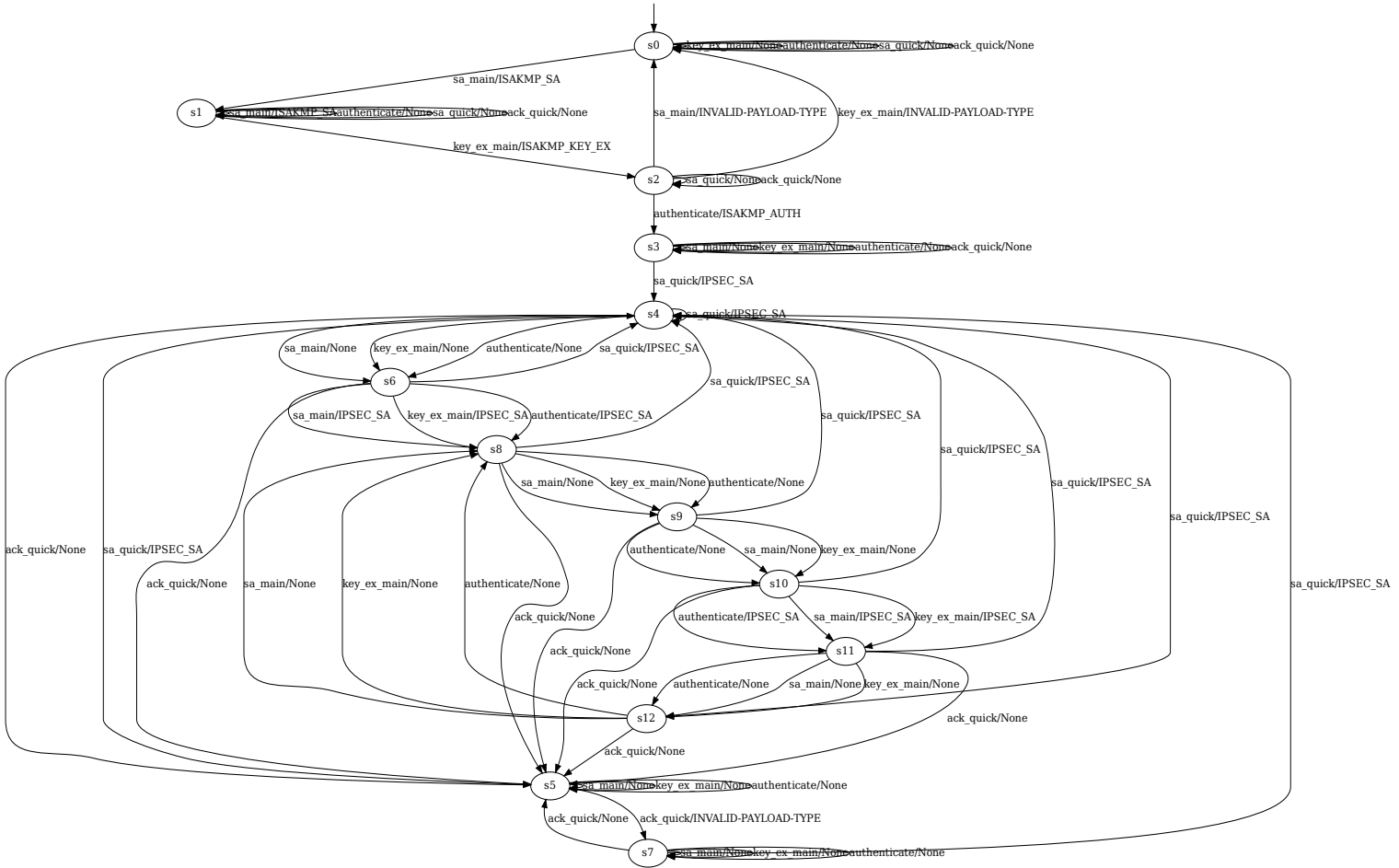
## References

1. Angluin, D.: Learning regular sets from queries and counterexamples. *Information and Computation* **75**(2), 87–106 (1987). [https://doi.org/10.1016/0890-5401\(87\)90052-6](https://doi.org/10.1016/0890-5401(87)90052-6)
2. Barker, E., Dang, Q., Frankel, S., Scarfone, K., Wouters, P.: Guide to IPsec VPNs. <https://doi.org/10.6028/NIST.SP.800-77r1>
3. Daniel, L.A., Poll, E., de Ruiter, J.: Inferring OpenVPN state machines using protocol state fuzzing. In: 2018 IEEE European Symposium On Security And Privacy Workshops (EuroS&PW). pp. 11–19. IEEE (2018)
4. Feldmann, A., Gasser, O., Lichtblau, F., Pujol, E., Poese, I., Dietzel, C., Wagner, D., Wichtlhuber, M., Tapiador, J., Vallina-Rodriguez, N., Hohlfeld, O., Smaragdakis, G.: A year in lockdown: How the waves of COVID-19 impact internet traffic. *Commun. ACM* **64**(7), 101–108 (2021). <https://doi.org/10.1145/3465212>
5. Ferguson, N., Schneier, B.: A cryptographic evaluation of IPsec (1999)
6. Fiterau-Brostean, P., Janssen, R., Vaandrager, F.W.: Combining model learning and model checking to analyze TCP implementations. In: Chaudhuri, S., Farzan, A. (eds.) *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II*. *Lecture Notes in Computer Science*, vol. 9780, pp. 454–471. Springer (2016). [https://doi.org/10.1007/978-3-319-41540-6\\_25](https://doi.org/10.1007/978-3-319-41540-6_25)
7. Fiterau-Brostean, P., Lenaerts, T., Poll, E., de Ruiter, J., Vaandrager, F.W., Verleg, P.: Model learning and model checking of SSH implementations. In: Erdogmus, H., Havelund, K. (eds.) *Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software, Santa Barbara, CA, USA, July 10-14, 2017*. pp. 142–151. ACM (2017). <https://doi.org/10.1145/3092282.3092289>
8. GmbH, A.C.V.: Connecting the FRITZ!Box with a company's VPN, <https://en.avm.de/service/vpn/connecting-the-fritzbox-with-a-companys-vpn-ipsec/>, accessed: 2023-03-03
9. Guo, J., Gu, C., Chen, X., Wei, F.: Model learning and model checking of ipsec implementations for internet of things. *IEEE Access* **7**, 171322–171332 (2019). <https://doi.org/10.1109/ACCESS.2019.2956062>
10. Harkins, D., Carrel, D.: The internet key exchange (IKE). RFC 2409, RFC Editor (11 1998), <https://www.rfc-editor.org/rfc/rfc2409.txt>
11. Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., Kivinen, T.: Internet key exchange protocol version 2 (IKEv2). RFC 7298, RFC Editor (10 2014), <https://www.rfc-editor.org/rfc/rfc7296.txt>
12. Kearns, M.J., Vazirani, U.V.: *An Introduction to Computational Learning Theory*. MIT Press (1994), <https://mitpress.mit.edu/books/introduction-computational-learning-theory>
13. Muskardin, E., Aichernig, B.K., Pill, I., Pferscher, A., Tappler, M.: Aalpy: an active automata learning library. *Innov. Syst. Softw. Eng.* **18**(3), 417–426 (2022). <https://doi.org/10.1007/s11334-022-00449-3>
14. Novickis, T., Poll, E., Altan, K.: Protocol state fuzzing of an OpenVPN. Ph.D. thesis, PhD thesis. MS thesis, Fac. Sci. Master Kerckhoffs Comput. Secur., Radboud Univ (2016)
15. Pferscher, A., Aichernig, B.K.: Fingerprinting Bluetooth Low Energy devices via active automata learning. In: Huisman, M., Pasareanu, C.S., Zhan, N. (eds.) *Formal Methods - 24th International Symposium, FM 2021, Virtual Event, November 20-26, 2021, Proceedings*. *Lecture Notes in Computer Science*, vol. 13047, pp. 524–542. Springer (2021). [https://doi.org/10.1007/978-3-030-90870-6\\_28](https://doi.org/10.1007/978-3-030-90870-6_28)



16. Pferscher, A., Aichernig, B.K.: Stateful black-box fuzzing of Bluetooth devices using automata learning. In: Deshmukh, J.V., Havelund, K., Perez, I. (eds.) NASA Formal Methods - 14th International Symposium, NFM 2022, Pasadena, CA, USA, May 24-27, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13260, pp. 373–392. Springer (2022). [https://doi.org/10.1007/978-3-031-06773-0\\_20](https://doi.org/10.1007/978-3-031-06773-0_20)
17. Pferscher, A., Wunderling, B.: Supplemental material “Mining Digital Twins of a VPN Server”, <https://doi.org/10.6084/m9.figshare.21953222.v1>, accessed: 2023-01-25
18. Rivest, R.L., Schapire, R.E.: Inference of finite automata using homing sequences. *Inf. Comput.* **103**(2), 299–347 (1993). <https://doi.org/10.1006/inco.1993.1021>
19. de Ruiter, J., Poll, E.: Protocol state fuzzing of TLS implementations. In: Jung, J., Holz, T. (eds.) 24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015. pp. 193–206. USENIX Association (2015), <https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/de-ruiter>
20. S, R.R., R, R., Moharir, M., G, S.: Scapy - a powerful interactive packet manipulation program. In: 2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS). pp. 1–5 (2018). <https://doi.org/10.1109/ICNEWS.2018.8903954>
21. Shahbaz, M., Groz, R.: Inferring Mealy machines. In: Cavalcanti, A., Dams, D. (eds.) FM 2009: Formal Methods, Second World Congress, Eindhoven, The Netherlands, November 2-6, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5850, pp. 207–222. Springer (2009). [https://doi.org/10.1007/978-3-642-05089-3\\_14](https://doi.org/10.1007/978-3-642-05089-3_14)
22. Stone, C.M., Chothia, T., de Ruiter, J.: Extending automated protocol state learning for the 802.11 4-way handshake. In: López, J., Zhou, J., Soriano, M. (eds.) Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11098, pp. 325–345. Springer (2018). [https://doi.org/10.1007/978-3-319-99073-6\\_16](https://doi.org/10.1007/978-3-319-99073-6_16)
23. Tappler, M., Aichernig, B.K., Bloem, R.: Model-based testing IoT communication via active automata learning. In: 2017 IEEE International Conference on Software Testing, Verification and Validation, ICST 2017, Tokyo, Japan, March 13-17, 2017. pp. 276–287. IEEE Computer Society (2017). <https://doi.org/10.1109/ICST.2017.32>

### Appendix A Learned Models



**Fig. 4.** Strongswan Server automata with retransmissions. The model took approximately 200 minutes (12187 seconds) to learn with the  $L^*$  algorithm, spread over five learning rounds. 631 membership queries and 130 equivalence queries were performed in 5188 and 1947 steps respectively. On average, three to four non-determinism errors were caught and fixed per learned model, arising from differing timings causing re-transmissions to be sent at different points.