Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

INSIDE DIGITAL TWINS: WHAT THEY ARE AND HOW THEY WORK

Steffen Strassburger

Group for Information Technology in Production and Logistics, Ilmenau University of Technology, Ilmenau, GERMANY

ABSTRACT

This tutorial gives an overview about the state-of-the-art of Digital Twins (DTs). We discuss what they are, what purposes they may fulfill, how they differ from related technologies, and how they may work internally. Application-wise, we focus on Digital Twins of production and logistics systems, but the majority of our explanations should also be valid beyond these domains. Finally, we discuss open issues and potential directions of future research.

1 INTRODUCTION

The term "Digital Twin (DT)" is used in a manifold of domains and contexts. It has many different definitions and interpretations. For the purposes of this tutorial, we focus on DTs of production and logistics systems. We adopt the definition developed in a meta review by Kuehner et al. (2021):

"A Digital Twin is a virtual representation of its physical counterpart. Its components provide the basis for a simulation or are simulation models themselves. The Digital Twin has an automated bidirectional data connection with the represented physical counterpart. This connection may span across several life phases of the system."

The advantage of this definition is that it is generic enough to embrace a majority of DT applications seen in literature (certainly beyond production and logistics), but also specific enough to distinguish DTs from related technologies and from plain simulation models as such. As a universally accepted definition of DTs is illusionary, we try to shift focus onto what DTs can be used for and what components and functions they must implement for a specific use case.

In principle, DTs can be used for many different purposes and with different objectives. The most common use cases found in literature can be divided into the following categories:

- Monitoring of the real system: The focus here is to provide an adequate digital representation of the real system and its current state. Potential implementations range from interactive web-based dashboards with key performance indicators up to high-resolution, real-time synchronized 3D-animations including detailed kinematics. A certain level of interactivity is often required.
- Prediction of future states of the real system: The focus here is on a faster-than-real-time prognosis of the future development of the real system based on its current state. This may serve as an early-warning-system, e.g., to give hints on potential upcoming problems, and help in scheduling decisions, e.g., on when to perform maintenance tasks. Applied methods typically include some form of simulation, but they can nowadays be challenged by or supplemented with data-driven methods, including machine learning.
- Control of the real system (direct or indirect): The focus here is on enabling the DT to perform some form of control of the real system. This will typically require some form of evaluation of the current state of the real system. It may include the simulation-based evaluation of different action alternatives and how future trajectories of system behavior would look like if a specific action is performed. Even a simulation-based optimization of a set of key performance indicators can be the basis for the control action. In the end, the DT suggests a certain control action to be taken (indirect

control) with a human verifying and implementing the control action, or executes the control action automatically (direct control).

Depending on the specific objectives of a use case, a DT must implement different components and provide different functionalities. It may be argued that only use cases involving control of the real system require a bidirectional data connection and thus qualify as a DT according to the definition given above. We will defer this discussion to section 2, where we discuss DTs and related technologies.

The rest of this tutorial is structured as follows: Section 3 discusses typical components and architectural approaches to DTs. In section 4, we will introduce common tasks and challenges in building a DT, and how they can be solved. In section 5, we will introduce some practical examples and discuss to which extent they qualify as DTs, and how they solve common implementation problems.

Section 6 concludes this paper and provides an outlook on unresolved issues and potential future research work needed.

2 DIGITAL TWINS AND RELATED TECHNOLOGIES

There have been several related technologies suggested in the simulation literature that need to be put into relation to what is now commonly referred to as a "Digital Twin". We base our discussion on Scheer et al. (2021), which focuses on digital-physical-interconnection concepts.

2.1 Offline Simulation

The classical application of simulation methods occurs in "offline" mode, i.e., there is no live data connection between the real system and its digital representation in a simulation model. This is obviously true if the simulation model is built in the planning phase of the real system. It is also true in many other use cases where the real system exists, and the simulation study follows a typical life cycle. Here, you will typically have a dedicated data collection phase, where relevant data of the real system is collected and probability distributions are fitted, if applicable. Only after data collection and model building, experiments with the simulation model are conducted. Simulation results are analyzed and may help decision makers in implementing system improvements.

2.2 Online Simulation

Online simulation (Davis 1998) introduced the first paradigm shift away from the traditional "offline" usage of simulation models. The basic idea in online simulations is that the simulation model should experience the same external influences as its physical counterpart. Also, an initialization of the simulation model based on the current state of the real system is typically a prerequisite to conduct online simulations (Hanisch et al. 2003, Bergmann et al. 2011). The behavior of an online simulation after it is initialized can differ: Some authors continuously update the simulation with observation data from the real system, others implement repeated cycles of simulation re-starts and online initializations. Both approaches have their individual advantages and associated challenges. In online simulations, there is no direct data flow from the simulation back to the real system. This behavior is summarized in Figure 1.

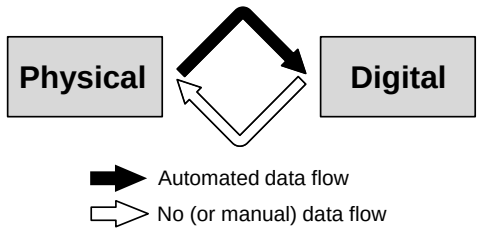

Figure 1: Online simulation requires some form of automated the data flow from the physical system to the digital representation.

2.3 Dynamic Data Driven Application Systems

Dynamic Data Driven Applications Systems (DDDAS) are a paradigm where measurement data from an operational system is dynamically incorporated into an executing model of that system, and computational results from the model are then used to guide the measurement process (Darema 2004). Measurement data is used by the model, e.g., to improve its accuracy, and computational results produced by the model help steer the measurement process itself. All control variables that a DDDAS can influence are purely virtual, i.e., the physical system is explicitly not influenced (Blasch et al. 2018).

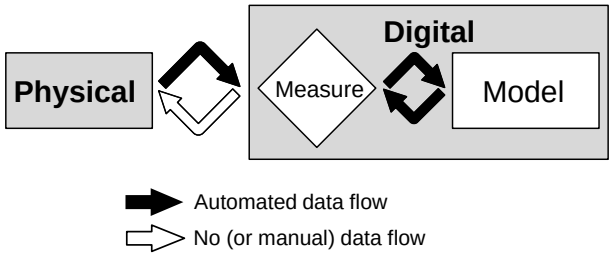

Figure 2: Data flows in dynamic data driven application systems.

2.4 Symbiotic Simulation

Symbiotic simulation refers to the coexistence of a simulation and a physical system that it represents (Aydt et al. 2008, Aydt et al. 2009). In this relationship, real-time data from the physical system is used to initialize and calibrate the simulation model. The physical system can also trigger what-if analyses and benefit from the simulation results. In symbiotic simulation open-loop and closed-loop forms are distinguished. In an open-loop system, only the simulation benefits from data of the real system. This behavior is similar to the concept of online simulation. In closed-loop systems, a bidirectional data exchange between the physical and the simulated system takes place. Figure 3 summarizes this behavior.

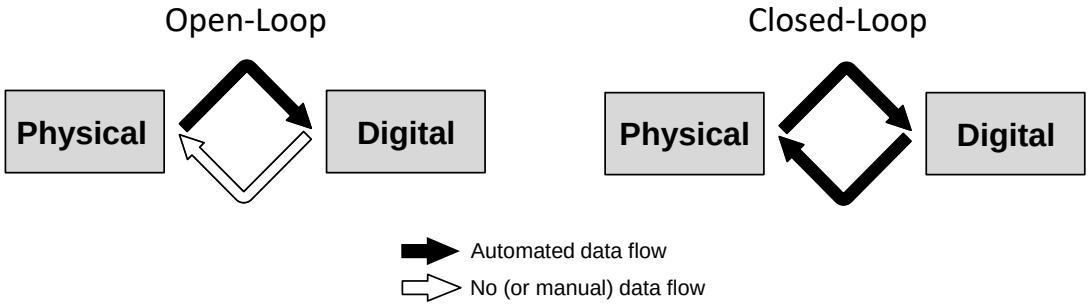

Figure 3: Data flows in symbiotic simulation.

2.5 Cyber-physical systems (CPS)

A Cyber-physical system (CPS) is an extension of a physical system with embedded computing capabilities (Lee 2015). It can monitor and control physical processes, mainly in the form of control loops. A CPS typically includes actuators and sensors for interacting with the environment. It also includes some form of digital representation of its environment and its state. In that sense, the digital representation is part of the CPS and there is a bidirectional data connection between the physical system and its digital representation.

The modeling capabilities of the cyber component can range in complexity from a differential equation describing behavior to a complex simulation model. However, the computing capabilities of a CPS are typically dimensioned towards the control task at hand. In that sense, simulation capabilities are likely limited to the most essential things for the intended control task.

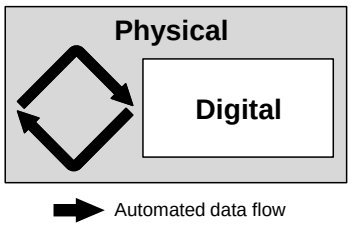

Figure 4: Data flows in cyber-physical systems.

2.6 Digital Twins

The origins of duplicating ("twinning") real objects can be traced back to the 1960s, where NASA created physical and digital copies of their moon rockets and space shuttles. The first modern age mentioning of the Digital Twin concept can probably be attributed to Michael Grieves in 2002, where he introduced a DT-like concept in a presentation about Product Lifecycle Management (Grieves 2023). In his concept, the virtual and real systems would be connected as the system went through the four phases of creation, production, operation, and disposal. We reflect the latter (a connection of a DT with its real counterpart can span across multiple life cycle phases) in our DT definition given in section 1.

In practice, many definitions of DTs exist. If you ask ChatGPT for a DT definition, it may answer something like the following:

"A **digital twin** is a virtual model of a physical object, system, or process that reflects its properties, behavior, and condition in real-time or near real-time. This digital representation is created and continuously updated through the collection and processing of sensor data as well as numerical simulations." (ChatGPT in March 2025, in the role of a scientist, when asked what a DT is).

Aside from being oddly specific by suggesting "numerical" simulations, this definition could well be a consensus to many in the field. So is there a point in elaborating further on the definition? From a practical standpoint, probably not. From an academic perspective, it makes sense to try to distinguish DTs from the related technologies discussed in the previous sections. This is achieved by our definition from section 1, which defines a DT by the following 4 points:

- It is a digital representation of the physical counterpart.
- It serves as a simulation basis or is itself a simulation model.
- It has a bidirectional (data) connection with the physical counterpart.
- The bidirectional connection can exist over several system life phases.

From this academic perspective, it makes sense to require a DT to have a bidirectional data connection as depicted in Figure 5.

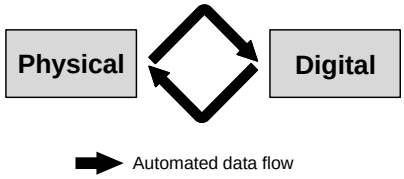

Figure 5: Data flows in Digital Twins.

This distinguishes DTs clearly from concepts such as online simulation and other related technologies. You can easily observe that the data flows are similar to those in the closed loop form of symbiotic simulation. Still, there is more to a DT than meets the eye of a simulationist. From an architectural point of view, a DT will typically involve more components and architectural considerations (see section 3) than a purely symbiotic simulation model.

From a practical perspective, many reported applications of DTs will not have a bidirectional data connection, especially the data or control flow back to the real system may be missing for a variety of reasons. Many practitioners will still argue that the term DT is correct if it represents a real system and its current condition in real time, even if there is no direct feedback.

This discrepancy between academia and practice is not necessarily a problem, as long as one is clear about the capabilities and objectives of a certain DT application. Still, using the term "Digital Twin" as an equivalent for traditional ("offline") simulation models is inadequate.

3 COMPONENTS AND ARCHITECTURES OF DIGITAL TWINS

The core components of a DT architecture that most will easily agree upon are the following:

1. **Physical object or system** – the real counterpart that is monitored or simulated
2. **Digital representation** – a virtual model that replicates the behavior and properties of the object
3. **Data connection** – enables the continuous exchange between the physical and digital twin.

But how should we put these components into an architecture, which connection technology should we use, how do we organize the data transfer and storage, and which additional components are needed?

While the discussion of specific implementation technologies is beyond the scope of this tutorial, we have analyzed typical architectural designs in DT literature (e.g. Uhlemann et al. 2017, Scheer et al. 2023a). Our common-sense approach of the components most frequently required and how they are typically put into an architectural context is shown in Figure 6.

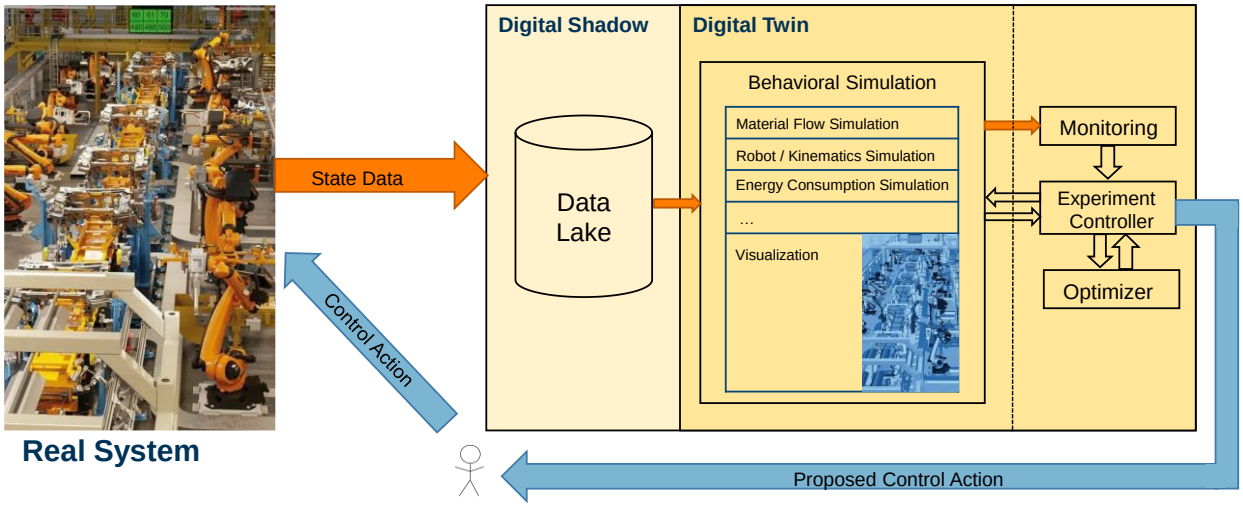

Figure 6: Typical DT components in a pragmatic DT architecture.

Obviously, every DT architecture will have some form of real system and some kind of data flow from the real system towards the digital side. This data flow may include any kind of observable state from the real system needed to mirror its state on the digital side. The data may originate from already existing sensors of automation equipment or other forms of production data acquisition. Some application scenarios

will require the installation of additional sensors/monitoring equipment. The most interesting data categories include the processing status of orders, the status/condition of queues and workstations, as well as the location of any movable equipment.

Modern companies often already have a basic data collection architecture in place, including a so called *data lake* as the target (and most often sink) of their data collection efforts. This data lake is depicted in our architecture. We refer to this data lake as a "Digital Shadow" (DS), a term we have so far avoided to use, as efforts to distinguish it from DTs may spark even more controversy.

Some authors define a DS by its automatic data flows from the real system to digital world (just like in online simulations), whereas a DT is said to also have an automatic data flow back to the real system (see Kritzinger et al. 2018). In our opinion, this is somewhat oversimplified. DS and DT also differ in functionality.

To us, a DS contains all the historic data collected from the real system needed to assess its current state. A DS is therefore capable of "shadowing" the current and past states of the real system, but it does not necessarily include any simulation (and thus forecasting) capabilities. This is a fundamental functional difference to DTs. From a data science perspective, one might argue that some forecasting capability might be easily included in a DS without actual simulation. While this could be a fair basis for further differentiation of digital shadows and digital twins, in our simulation-centric architecture, we limit the function of the DS to its data storage capabilities.

Independent from this terminology, many DT architectures will have some intermediate data storage capability for holding incoming state data from the real system. Note that in practical environments data lakes may have significant delays (up to several minutes) for reflecting collected data. Use cases without an intermediate data storage would feed data from the real system directly into a simulation. In our estimation, this is rather rare in practical environments.

The state data contained in the digital shadow is the base for reflecting the behavior of the real system in the behavioral simulation component of the DT. Different kinds of simulation may be required depending on the objectives of the DT use case. Strategies for initializing the simulation and maintaining synchronicity between the simulation and the real system are discussed in section 4.2. The main purpose of the simulation is to interpolate and extrapolate system state based on the last known condition of the real system. The interpolation typically bridges the gap between two state updates from the real system. Here, the simulation is typically executed in a real-time-synchronized fashion with the real system. It will typically include easily observable visualizations of the evolving state of the real system. In the extrapolation mode, the simulation typically evaluates multiple what-if-scenarios in fast-forward mode.

For the latter purposes, some additional components will typically be required, that may or may not be considered to be part of the DT (indicated by the dashed line in Figure 6). These components include a monitoring component that detects or cyclically checks the need for and the potential of control actions. The need for a control action may be detected by monitoring certain key performance indicators and their development (e.g. decreasing throughput, increasing queue contents, increasing backlog). The monitoring may already be combined with repeated fast-forward simulation executions. When the need for a control action is detected, multiple (typically pre-defined) actions can be evaluated. This may even include optimization approaches. In the end, an action can be proposed and initiated. In practical scenarios, a human decision maker will typically be included in this control loop as a safeguard, but this is not necessarily the case.

4 TYPICAL ISSUES IN BUILDING A DIGITAL TWIN

After having discussed major components and a potential DT architecture, we now continue to discuss typical problems and design choices that will be encountered when implementing a DT. In this year's edition of the tutorial we focus on data acquisition and simulation model generation, initialization, and execution modes. Other typical issues, not discussed due to time and space constraints, include the design of appropriate DT user interfaces, experimentation and optimization features, condition monitoring and anomaly detection as well as system control.

4.1 Data Acquisition

In our experience many DTs for production and logistics systems will be built upon an existing data storage and communication infrastructure. Your data connection to the real system will therefore likely depend on which interfaces and protocols your real systems offers. Modern architectures have a trend towards efficient protocols such as OPC/UA and MQTT. Often, there will already be some form of data lake in place that your DT is expected to frugally feed itself from. In this case you need to know and critically evaluate the time delay that such an architecture entails. A close-to-real-time monitoring may be out of the question if delays are in the area of multiple minutes. Other DT applications, such as cyclic performance predictions and evaluations, may have no problems with such delays.

If a data storage infrastructure can be built from scratch, it is advisable to look into standardized factory data formats, such as the Core Manufacturing Data Format (Bergmann and Strassburger 2015). For certain DT applications it may even be sufficient to use that data storage capabilities that your simulation system of choice offers.

4.2 Simulation Model Generation, Initialization and Execution Modes

Next to handling data acquisition, you obviously need to think about your simulation strategy. The type of simulation required depends on the objectives of your DT. For many objectives in production and logistics, a (material flow) simulation model based on some discrete event simulation paradigm will be a good choice. Many practitioners base the simulation part of their DT environment on a commercial-off-the-shelf (COTS) simulation system. Many COTS vendors even advertise their simulation systems as being a good base for developing DTs. While this may be true in varying extent, do be careful about the proper use of the term DT. Not every simulation model is a Digital Twin even if advertised so. Some practitioners also rely on open-source simulation libraries, such as SimPy, as the base for the simulation capabilities of their DT (Scheer et al. 2023b).

Next to the choice of simulation environment, one needs to decide about the proper approach for simulation model generation: Is this a one-time task that is best done manually, or is it better to apply some form of automatic model generation (AMG)? The latter is certainly advisable, if changes to the real system occur frequently and in a way that requires the model to be rebuilt from scratch. Here, you need to think about an efficient way to implement AMG and take into account its well-known challenges (Bergmann and Strassburger 2010).

If the structure of the real system remains unchanged for a longer period, manually building your model may be the decision of choice. Still, even in such cases, you need to think about model adaptability and how to reflect gradual changes in your real system (e.g., decreasing performance of your equipment, effect on processing times, changing probability distributions).

Next to the strategy for simulation model creation, simulation model initialization needs to be decided upon and implemented. This is essential for initially synchronizing your "empty" DT with the live state of your real system. In case of an AMG approach, model generation and initialization can often be combined (Bergmann et al. 2011). In case of a manual model generation, an appropriate way of accessing the state of the real system based on data from the Digital Shadow, and reflecting that state in the simulation model is needed. The concrete implementation depends on the use case and its available data. Sometimes educated guesses will need to be taken, e.g., if exact positions of movable equipment are not immediately available, or if the remaining processing time of currently started jobs is unknown. Some "phasing in" strategy may be required that adjusts the initial DT state with subsequent updates of the real system.

Now, let's assume that your DT is initialized correctly and reflects the current state of your real system. Where do we go from here, i.e., which simulation execution modes do we need? This highly depends on your objectives. We briefly discuss the categories introduced in section 1.

If your main objective is to have the DT as a *monitoring* instrument, the main task of your simulation is likely to interpolate and visualize the state of the real system between state updates received via the

Digital Shadow. Hence, your simulation will need to run in a real-time synchronized fashion and incorporate state updates from the real system whenever they are available to improve its accuracy.

Beyond that, more tricky questions arise. Let us consider the treatment of stochastic influences such as machine failures or stochastically distributed processing times. Normally, such stochastic influences would be modelled by a suitable probability distribution. But do you really want your *monitoring* DT to create and indicate a machine failure when there is none in the real system? Most likely not. You would likely want the real system to introduce such events if and when they actually occur. But how do you deal with stochastic variation in processing times? For some systems the best choice will be to model a processing time by its mean value (although this would normally constitute one of the famous pitfalls of simulation modeling (Law 2024)). The reason for this recommendation is that this will likely match the behavior of your real system as closely as possible. Additionally, the forecast horizon in a monitoring DT will likely be rather small, thus the divergence between DT and real system may be minimized. Obviously, you will need to correct your DT state when any contradicting input from the real system is received. A viable approach may be to phase in any required corrections. In our experience, many practitioners report that their DTs (at least for short term time horizons) exhibit completely deterministic behavior. The treatment of stochastic influences in the prediction and control scenarios discussed below depends on your specific use case and the forecast horizon.

If your main objective is to have the DT for *prediction* purposes, the main task of your DT is to enable a faster-than-real-time prognosis of the future development of the real system based on its current state. Hence, your simulation will need to run in an as-fast-as-possible execution mode. A typical solution for this use case is to implement repeated simulation initialization and prediction cycles.

Lastly, in DTs for *control* purposes, the dominating execution mode of your simulation could either be in a real-time synchronized fashion or in an as-fast-as-possible mode, or a combination of both. The first choice would initiate control actions once an anomaly is detected at the current time of the real system (e.g., when a threshold of a KPI is exceeded), the latter choices would enable repeated predictions of the influence of control actions using multiple what-if-scenario executions, or even simulation-based optimizations.

In practice, there may not always be a strict separation between the different use cases, i.e., there may be combinations of DTs for monitoring, prediction, and control. Such scenarios are often based on a monitoring DT that needs to maintain synchronicity with the real system, and a cloning capability (Schulze et al. 2000) to create identical copies of the monitoring DT's simulation. The cloned copies will then be the basis to perform additional as-fast-as-possible evaluations. When using COTS simulation packages, efficient cloning capabilities and multiple parallel simulation run options may not be available out-of-the-box or be subject to separate license agreements.

5 PRACTICAL EXAMPLES

In this section, we illustrate two practical DT examples and how their implementation solved the issues discussed in sections 3 and 4.

5.1 Digital Twin for Monitoring of a Streetcar System

The streetcar system in the city of Magdeburg was the earliest DT-like simulation project that the author was involved in (Schulze et al. 1999a, 1999b). The real streetcar system consisted of about 10 streetcar lines with an extensive route network across the city. At certain key locations of the real system, fixed sensors were installed that collected information of passing streetcars (timestamps and IDs), hence enabling a basic location tracking capability. (Remember – this was the late 1990s, before GPS and cell communication became ubiquitous.)

This system was mirrored in a simulation model which, at its core, was able to perform a schedule based simulation of the streetcar traffic and visualize the current positions of the streetcars according to its schedule over a simulation time of 24h. The simulation system was able to incorporate real-time location data received from the real streetcar system and adjust its current state according to the data received, i.e.,

the simulated position of streetcars was updated to their actual position, and simulation continued based on this actual position.

In today's terminology, we created a classical monitoring DT, which was able to visualize the current positions of all streetcars in close-to-real time. In between the sparse location updates, the simulation would do its best to interpolate system state and update and visualize current location of streetcars as closely as possible (see Figure 7). There was no control feedback towards the streetcar system (i.e. no bidirectional data connection), so the argument might be made whether the term DT is adequate. At the time of its creation, it was called online simulation. Following our own strict DT definition, this would not qualify as a DT, but in layman's terms, this is as close to a DT as it gets.

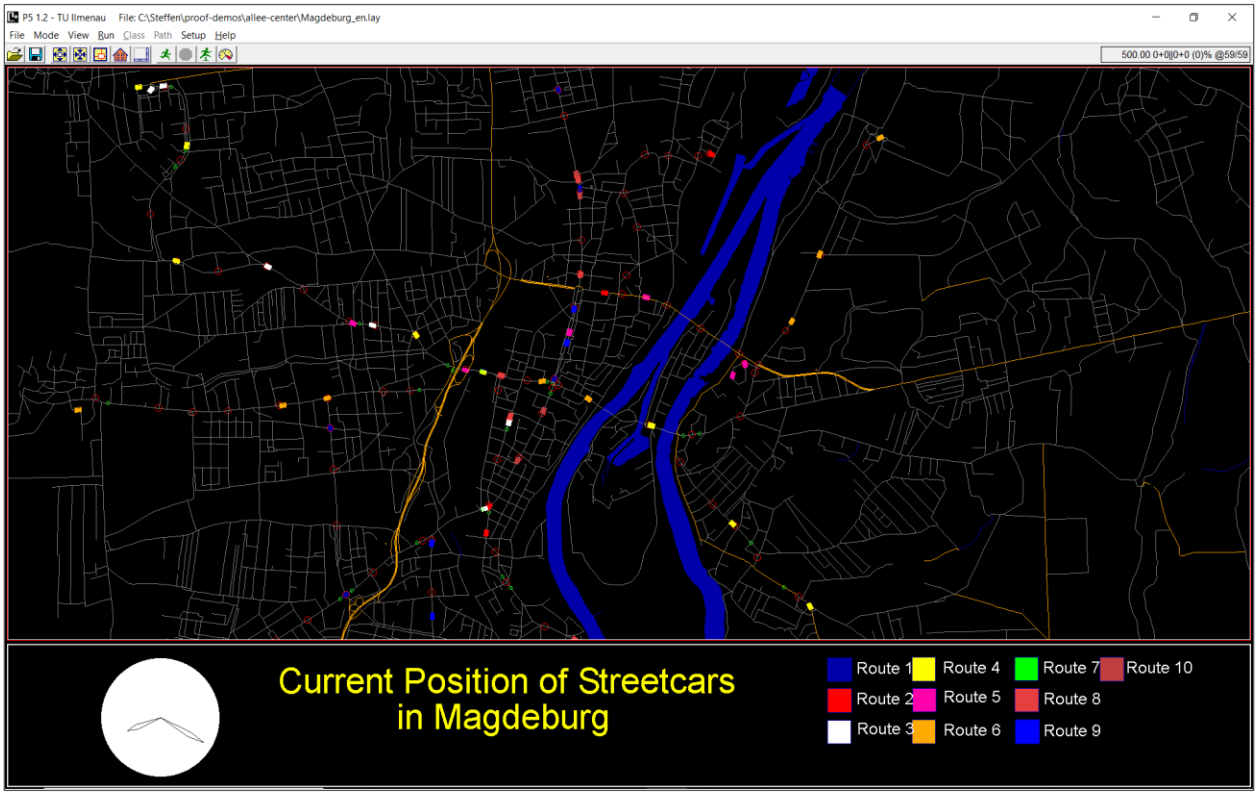

Figure 7: Visualization of current streetcar positions based on an online simulation of the real streetcar system - a monitoring DT's visualization in today's terminology.

How did we solve the implementation problems discussed in the previous sections? Upon simulation start, the simulation (implemented in SLX and Proof Animation) would fast-forward simulate to the current wall-clock time based on the streetcar schedule. From that time onward, a real-time synchronicity would be maintained. The location data from the real system was collected in a central control system of the streetcar operator. Using a dial-up modem connection, our simulation was able to obtain any location data collected after the connection was established. There was no intermediate data storage in the sense of a Digital Shadow. Real-time data was then fed directly into the simulation model and processed immediately. Data telegrams had delays of about 30 seconds. That delay was taken into account accordingly when adjusting streetcar positions. There were no stochastic influences modeled in the streetcar simulation. The system worked well (within the stability limits of dial-up connections) and was demonstrated live at the 1999 Winter Simulation Conference (Schulze et al.1999b).

5.2 Digital Twin of an Existing Manufacturing Line

In (Scheer et al. 2023a) and (Scheer et al. 2023b) we experimented with the best pragmatic approach for the creation of DTs for existing real-life manufacturing lines. We recommend pragmatic choices for including existing infrastructure as components of a DT. When an existing data collection infrastructure is in place (in our case it was called "data lake"), we suggest to use it instead of trying to build a new one. Then feed your DT frugally from this data lake and try to make the best of what data is already collected. It is the path of least resistance from the stakeholders you need to involve. Based on the historic data in this data lake, often certain data pre-processing steps can be taken that are beneficial for initialization purposes of the DT. In our case, we experimented with certain data preprocessing methods to get the most information for initialization of our DT. We therefore labeled our approach as "hybrid".

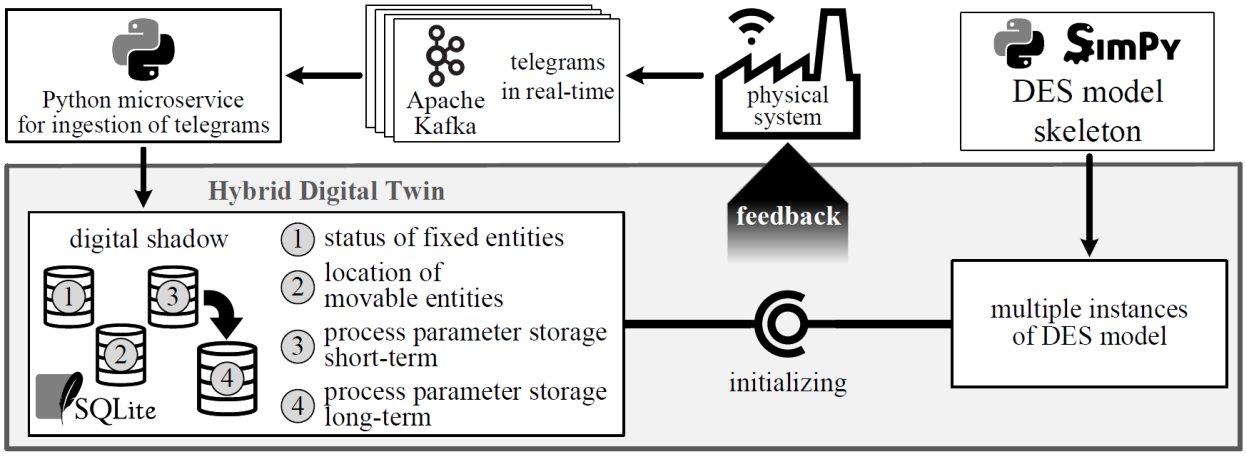

Figure 8: Architecture for a "hybrid" DT of manufacturing lines, applied to a real-life manufacturing line of a large automotive supplier in Germany (Scheer et al. 2023a).

In the end, a DT for a specific manufacturing line was implemented and successfully tested. The simulation component was implemented in SimPy. The core simulation model was manually created. Model initialization was performed automatically using data from the data lake. As our main use case included short- to mid-term predictions, we decided for a cyclic restart and initialization of the simulation component (instead of constantly updating it). We then conducted fast-forward simulations. Certain stochastic influences such as machine break-downs were intentionally excluded from the simulation model. The main reason for this was the intended use case of making short- to mid-term predictions of the performance of the real system under current work load and potentially suggesting control recommendations towards the physical system. Potential equipment breakdown was not a factor that was useful to be taken into account for this use case, as it would have created ambiguity in the recommendation of the control action. Obviously, there can be other use cases for DTs, where stochastic equipment failure needs to be modeled (e.g., for purposes of predictive maintenance, or for the evaluation of equipment failure avoidance or reaction strategies).

6 SUMMARY AND OUTLOOK

This tutorial has discussed what is nowadays considered as a DTs, how they may work internally, and what typical issues you will encounter when trying to implement one. We have tried to give some guidance on distinguishing DTs from related concepts. Keep in mind that this is an evolving topic and that the term DT tends to get used inflationary, sometimes also for things such as purely offline simulation models. So be

prepared to look behind the buzzword, ask the right questions, and also call out inadequate usage of the term.

In this tutorial we have only covered the issue of single DTs, e.g., we discussed how to build one dedicated DT for one dedicated manufacturing system. In practice, we will always be confronted with systems that contain other systems, e.g., a car factory containing different production areas (body in white, paint shop, final assembly) that again contain smaller systems such as machines and robots. When each system in a system-of-systems can have its own DT, in future we will also need approaches for building hierarchies of DTs and for combining (or federating) multiple distributed DTs. This is a rather unexplored research area that will require more work. See (Kunnari and Strassburger 2025) for a state-of-the-art discussion.

Another topic that requires more research is the connection of real systems with their DTs. Interoperability solutions either look at facilitating interoperability between real equipment (such as OPC/UA for automation components) or at facilitating interoperability between simulation systems (such as the High Level Architecture for Modeling and Simulation (HLA), now in its 2025 version). There is still no standard that is equally suited for addressing the need of both types of systems (Strassburger, 2019).

The future will also show if ideas such as the *asset administration shell* (put forward by Industry 4.0-stakeholders) that has long been promoted as a DT-like representation of physical assets, will finally materialize and gain traction.

REFERENCES

Aydt, H., S. Turner, W. Cai, and M.Y.H. Low. 2008. “Symbiotic Simulation Systems: An Extended Definition Motivated by Symbiosis in Biology”. In *22nd International Workshop on Principles of Advanced and Distributed Simulation*, June 3rd-6th, Rome, Italy, 109-116.

Aydt, H., S. Turner, W. Cai, and M.Y.H. Low. 2009. “Research issues in symbiotic simulation”. In *2009 Winter Simulation Conference (WSC)*, 1213-1222 https://doi.org/10.1109/WSC.2009.5429419.

Bergmann, S. and S. Strassburger. 2010. “Challenges for the Automatic Generation of Simulation Models for Production Systems”. In *Proceedings of the 2010 Summer Simulation Multiconference*, July 11th-14th, Ottawa, Canada, 545-549.

Bergmann, S., S. Stelzer, and S. Straßburger. 2011. “Initialization of Simulation Models Using CMSD”. In *2011 Winter Simulation Conference (WSC)*, 2228-2239 https://doi.org/10.1109/WSC.2011.6147934.

Bergmann, S. and S. Strassburger. 2015. “On the Use of the Core Manufacturing Simulation Data (CMSD) Standard: Experiences and Recommendations”. In *Proceedings of the 2015 Fall Simulation Interoperability Workshop*. August 30th - September 4th, Orlando, USA.

Blasch, E., D. Bernstein, and M. Rangaswamy. 2018. “Introduction to Dynamic Data Driven Application Systems”. In *Handbook of Dynamic Data Driven Application Systems*, edited by E. Blasch, S. Ravela, and A. Aved, 1-25. Cham: Springer.

Darema, F. 2004. “Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements”. In *Proceedings of the International Conference on Computational Science*, 662–669. Berlin, Heidelberg: Springer.

Davis, W. J. 1998. “On-Line Simulation: Need and Evolving Research Requirements”. In *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, edited by J. Banks, 465-516. New York: John Wiley & Sons.

Grieves, M.W. 2023. “Digital Twins: Past, Present, and Future”. In *The Digital Twin*, edited by N. Crespi, A.T. Drobot, R. Minerva, 97-121. Cham: Springer.

Hanisch, A., J. Tolujew, U. Raape, and T. Schulze. 2003. “Online-Simulation für Personenströme in einem Frühwarnsystem“. In *Proceedings of the 17th Symposium Simulationstechnik (ASIM 2003)*, September 16th-19th, Magdeburg, Germany, 221-226.

Kritzinger, W., M. Karner, G. Traar, J. Henjes, and W. Sihn. 2018. “Digital Twin in manufacturing: A categorical literature review and classification”. *IFAC-PapersOnLine* 51(11): 1016-1022 https://doi.org/10.1016/j.ifacol.2018.08.474.

Kuehner, K., R. Scheer, and S. Strassburger. 2021. “Digital Twin: Finding Common Ground - A Meta-Review”. *Procedia CIRP* 104: 1227-1232 https://doi.org/10.1016/j.procir.2021.11.206.

Kunnari, A. and S. Strassburger. 2025. “Distributed, Hierarchical Digital Twins: State-of-the-Art, Challenges and Potential Solutions”. Accepted for: *2025 Winter Simulation Conference (WSC).*

Law, A. 2024. “Twenty-Three Critical Pitfalls in Simulation Modeling and How to Avoid Them”. In *2024 Winter Simulation Conference (WSC)*, 1296-1307 https://dl.acm.org/doi/10.5555/3712729.3712837.

Lee, E. A. 2015. “The past, present and future of cyber-physical systems: A focus on models”. *Sensors* 15(3): 4837-4869.

Scheer, R., S. Straßburger, and M. Knapp. 2021. “Digital-physische Verbundkonzepte: Gegenüberstellung, Nutzeffekte und kritische Hürden“. In *Proceedings of the 19th ASIM Dedicated Conference on Simulation in Production und Logistics*, September 15th-17th, Erlangen, Germany, 11-20.

Scheer, R., S. Strassburger, and M. Knapp. 2023a. “Hybridization of the Digital Twin - Overcoming Implementation Challenges”. In *Proceedings of the 56th Annual Hawaii International Conference on System Sciences (HICSS 2023)*, January 3rd-7th, Lahaina, HI, USA, 1438-1447.

Scheer, R., S. Straßburger, and M. Knapp. 2023b. “Der Hybride Digitale Zwilling: eine praxistaugliche Verbindung von Simulation und operationellen Produktionssystemen”. In *Proceedings der 20. ASIM-Fachtagung "Simulation in Produktion und Logistik"*. September 13th-15th, Ilmenau, Germany, 91-101.

Schulze, T., S. Straßburger, and U. Klein. 1999a. “On-line Data Processing in a Civil Transportation Federation”. In *Proceedings of the 1999 Fall Simulation Interoperability Workshop*. September 12th-17th. Orlando, USA.

Schulze, T., S. Straßburger, and U. Klein. 1999b. “On-line Data Processing in Simulation Models: New Approaches and Possibilities through HLA”. In *1999 Winter Simulation Conference (WSC)*, 1602-1609 https://doi.org/10.1109/WSC.1999.816899.

Schulze, T., S. Straßburger, and U. Klein. 2000. “HLA-Federate Reproduction Procedures in Public Transportation Federations”. In *Proceedings of the 2000 Summer Computer Simulation Conference*, July 16th-20th, Vancouver, Canada.

Strassburger, S. 2019. “On the Role of Simulation and Simulation Standards in Industry 4.0”. In *Proceedings of the 2019 Simulation Innovation Workshop (SIW)*. February 11th – 15th, Orlando, FL, USA.

Uhlemann, T., C. Lehmann, and R. Steinhilper. 2017. “The Digital Twin: Realizing the Cyber-Physical Production System for Industry 4.0”. *Procedia CIRP* 61: 335 – 340 https://doi.org/10.1016/j.procir.2016.11.152.

AUTHOR BIOGRAPHY

STEFFEN STRASSBURGER is a professor at the Ilmenau University of Technology and head of the Group for Information Technology in Production and Logistics. Previously he was head of the "Virtual Development" department at the Fraunhofer Institute in Magdeburg, Germany and a researcher at the DaimlerChrysler Research Center in Ulm, Germany. He has been involved with simulation for more than 25 years. He holds a Doctoral and a Diploma degree in Computer Science from the University of Magdeburg, Germany. His research interests include distributed simulation, automatic simulation model generation, and general interoperability topics within the digital factory and Industry 4.0 context. His email address is steffen.strassburger@tu-ilmenau.de. The website of his department is https://www.tu-ilmenau.de/itpl/.