Automatic Verification and Validation of Automatically Generated Simulation-Based Digital Twins for Discrete Material Flow Systems

---

Author: Daniel Fischer

Supervisor: Prof. Christian Schwede

Program: Research Master Data Science

Institution: Hochschule Bielefeld (HSBI)

Submission Date: April 3, 2025

Abstract

This thesis investigates the application of Digital Twins within discrete material flow systems, a key component of Industry 4.0. It reviews the progression from Digital Models and Digital Shadows to fully realized Digital Twins that integrate real-time data, simulation, and control mechanisms. To learn these Digital Twins from data enables companies to reduce twin creation and updating efforts. These gained advantages would be made obsolete if the validation and verification of such twins were performed manually involving experts. To address this problem, this research proposes a data-driven framework for automated verification, validation, and uncertainty quantification of simulation-based digital twins. Leveraging object-centric event logs and advanced machine learning techniques, the framework aims to seamlessly integrate digital simulation with real-world data. This approach ensures that the resulting Digital Twins maintain high fidelity and robustness by continuously benchmarking their performance against operational metrics. The framework is evaluated via a comprehensive case study examining its effectiveness in enhancing process efficiency and predictive precision.

Keywords: Digital Twins, Automated VVUQ, Process Mining, Machine Learning, Discrete Material Flow Systems.

Contents

List of Abbreviations

AGDT Automatically Generated Digital Twin. 1

ASMG Automatic Simulation Model Generation. 1

BNN Bayesian Neural Network. 1

BPMN Business Process Model and Notation. 1

CNN Convolutional Neural Network. 1

CPS Cyber-Physical System. 1

DDDT Data-Driven Digital Twin. 1

DE Deep Ensembles. 1

DES Discrete-Event Simulation. 1

DM Digital Model. 1

DMFS Discrete Material Flow Systems. 1

DS Digital Shadow. 1

DT Digital Twin. 1

E2O Event-to-Object Relation. 1

GP Gaussian Processes. 1

HDT Hybrid Digital Twin. 1

IoT Internet of Things. 1

KPI Key Performance Indicator. 1

LSTM Long Short-Term Memory. 1

MCD Monte Carlo Dropout. 1

ML Machine Learning. 1

O2O Object-to-Object Relation. 1

OCEL Object-Centric Event Log. 1

PM Process Mining. 1

PPC Production Planning and Control. 1

RNN Recurrent Neural Network. 1

SBDT Simulation-Based Digital Twin. 1

V&V Verification and Validation. 1

VVUQ Verification, Validation, and Uncertainty Quantification. 1

XES eXtensible Event Stream. 1

Chapter 1

Introduction

1.1 Initial Situation

Digital Twins (DT) are a key technology at the front of the fourth industrial revolution, often referred to as Industry 4.0. The latter term is characterized by the integration of cyber-physical systems (CPS), the Internet of Things (IoT), and cloud computing to create smart factories aimed at automation and efficiency (**Oztemel2020**). Companies pursue this vision by trying to remain competitive through the adoption of innovative technologies that promise enhanced productivity and reduced operational costs. One such technology that supports this transformation is the DT. It can be defined as a virtual representation of physical assets enabling real-time monitoring and optimization (**Tao2018ijamt**). The DT bridges the connection between the two entities with a bi-directional data flow to exchange information and to influence the behavior of the physical asset (**grieves2014digital**). This technology is central to Industry 4.0, facilitating the physical and digital worlds through real-time data integration, simulation, and optimization (**judijanto2024trends**).

Although this discipline is rapidly evolving, a unified definition of DT has yet to be established due to the diverse requirements and perspectives across different fields. In engineering, the focus might be on the real-time interaction between physical systems and their digital counterparts, whereas in computer science, the emphasis is often on data integration and simulation capabilities. These varying priorities result in multiple interpretations and applications of the term DT. The concept was first introduced by Michael Grieves in 2002, who defined it as a digital representation of a physical object or system (**grieves2014digital**). However, the concept has evolved since, encompassing a broader range of applications and

technologies. In the literature, three termes are used to describe similar characteristics of DT: Digital Model (DM), Digital Shadow (DS), and Digital Twin (DT), see Figure 1.1 (**jones2020characterising**; **Zhang2021jmsy**).

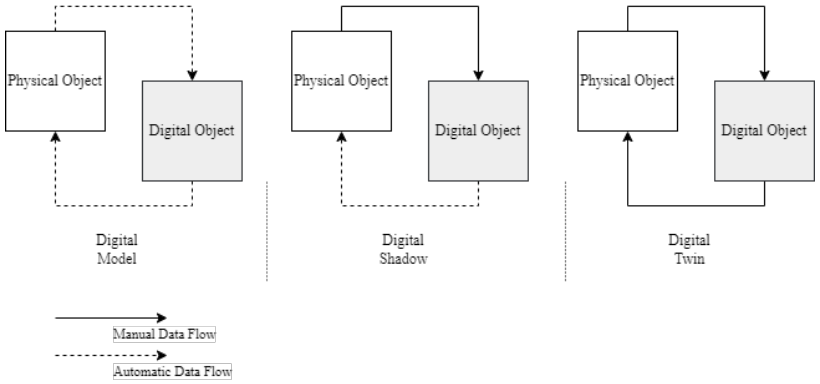

Figure 1.1: Comparison of Digital Shadow (DS), Digital Model (DM) and Digital Twin (DT) as presented by Kritzinger (2018). This distinction is crucial for understanding validation requirements across different digital representation types.

The Digital Model (DM) represents the most basic form. It involves manual data connections between physical and digital entities. These connections can be temporarily shifted or even disconnected. There is no direct control of the digital object over the physical entity. It is primarily a simple or complex model *describing* the physical object. The data flow must be manually triggered by the modeler, who also interprets the results and controls the DM. The Digital Shadow (DS) is a more advanced version of the DM. It is a digital representation of the physical object that is continuously updated with real-time data, allowing for monitoring, analysis and simulation. While it can predict future states of the physical object based on the current state and historical data, it is not able to influence the physical object without human intervention. The control is, similar to the DM, still in the hands of the modeller. A DS is frequently used for simulation purposes and is sometimes misclassified as a DT in the literature (**kritzinger2018digital**; **sepasgozar2021differentiating**). The Digital Twin (DT) is the most advanced version of the three, offering a digital representation of the physical object, which is also continuously updated with real-time data. The DT can be used for monitoring, analysis, and *control* purposes. It can predict the future states of the physical object based on the current state and historical data. The DT can also influence the physical object by sending control signals to it. The control is partially or completely in the hands of the DT. The DT thus *can* serve more purpose than modelling or simulating the physical object. It may serve as an autonomous system, updating itself or with minimal human intervention (**kritzinger2018digital**).

DTs are applied across various sectors, including manufacturing, defense, automotive, service, finance and healthcare (**Tao2018ijamt**). Manufacturing is particularly notabl due to its high potential for process optimization and automation. This thesis focuses on the latter, particularly discrete material flow systems (DMFS). These systems process discrete objects (parts) moving along transportation routes or conveyor lines at regular or irregular intervals, integrating both production and logistics operations (**arnold2005materialfluss**; **schwede2024learning**). A key simplification in their modeling is the abstraction of material flow as a sequence of discrete events, following the principles of discrete-event simulation (DES) (**kovacs2016mathematical**; **robinson2014simulation**). DES is well-suited for analyzing complex systems where state changes occur at discrete points in time, such as arrivals, departures, and processing steps (**robinson2014simulation**).

Historically, DM played a crucial role in the design, planning, and control of DMFS, primarily through applications like material flow simulations, logistic assistance systems, and digital factory implementations (**Thiede2013**). However, advancements in both DS and DT have enabled a shift from isolated, use-case-specific models toward complete digital representations that span the entire lifecycle of DMFS (**Abdoune2023**). This transition is largely driven by the growing demand for predictive capabilities by stakeholders and automated decision support in manufacturing systems, reflecting the core principles of Industry 4.0 (**frank2019industry**). A second driver of DT innovation lies in the widely available data from IoT devices and sensors, which enhances model training and real-time adaptation of DTs (**Tao2018ijamt**).

In practice, the automated data transfer between the digital model and the physical system is not always critical for DMFS management. Unlike in time-sensitive applications, human decision-makers often remain integral to the control loop, meaning that real-time automation is not always necessary (**schwede2024learning**). Therefore, for this thesis, DS and DTs will be treated as equivalent concepts.

Beyond replicating the current state and managing historical data, DTs are essential for predicting system behavior and evaluating potential modifications. The widespread use of DES within digital twins highlights the central role of simulation-based DTs (SBDTs) in DMFS (**Lugaresi2021aifac**). As **schwede2024learning** emphasize, SBDTs provide decision support for optimizing costs and performance in highly competitive manufacturing environments. While current SBDTs are primarily developed and updated manually by domain experts, emerging research explores how machine learning (ML) can enhance predictive accuracy and automate model updates by automatically learning model characteristics, re-

ducing costs and development time.

Thus, the progression from digital models to simulation-based DTs reflects an ongoing shift toward data-driven, predictive, and increasingly automated representations of DMFS, enabling more informed decision-making throughout the system's lifecycle (**boschert2016digital**; **lim2020state**).

1.2 Problem

Despite the transformative potential of DTs, their implementation can be challenging. Creating and maintaining accurate DTs require substantial investments in technology and domain knowledge. This investment is wasted if the resulting model fails to accurately represent the physical entity or produces incorrect results. While automatic generation may seem like an elegant solution, it carries risks such as overfitting or biased predictions (**gemanbias**). Manufacturing data for training must be rigorously cleaned and preprocessed. Automatically generated DTs must also undergo automatic Validation, Verification, and Uncertainty Quantification (VVUQ) to preserve their cost and time advantages. Manual VVUQ, which relies on humans in the loop, hinders scalability, automatic synchronization with the physical entity, and depends on costly domain knowledge often provided by experts (**Bitencourt2023**). These hurdles are significant barriers to automatic learning (**ribeiro2016should**; **zhao2024data**). As industries integrate DT into their production processes, establishing trust becomes fundamental as well (**trauer2022digital**; **arrieta2020explainable**). For widespread acceptance among coworkers, stakeholders, and investors, automatic DT creation and VVUQ must demonstrate clear advantages over manual creation and expert-led VVUQ.

Even when DT learning is successfully performed, questions about its correctness, precision, and robustness persist. These concerns are addressed by validation, verification, and uncertainty quantification frameworks (VVUQ) (**sel2025survey**). Ensuring the validity, reliability, and accuracy of a DT is critical, yet traditional VVUQ approaches rely heavily on manual expert involvement and case-specific reference values (**Bitencourt2023**; **hua2022validation**). This leads to inefficiencies, particularly in the context of automated DT generation, where such manual processes undermine the goal of reducing development effort. **hua2022validation** even argue that there are no robust and standardized verification and validation methods for DTs. As (**sel2025survey**) point out, uncertainty quantification is often overlooked, but addresses an important aspect of assessing low noise in explanations. One hurdle to standardized VVUQ frameworks is the lack of a clear

definitions for validity and verification in the context of DTs (**Bitencourt2023**).

For discrete material flow systems (DMFS), these challenges are even more pressing due to their procedural nature and inherent stochasticity. Rigorous VVUQ is essential to address the risk of manufacturing process failures caused by anomalies, resource constraints, software faults, or human error. This necessity arises because such failures can disrupt the intricate workflows and unpredictable dynamics inherent in DMFS, making reliable performance prediction a priority. When DTs for these systems are generated automatically, traditional validation methods become problematic, as they negate much of the efficiency gains through automation. This creates a fundamental conflict: while automated DT generation reduces initial development and updating efforts, it simultaneously increases the complexity of validation and verification, potentially counteracting its intended efficiency gains.

1.3 Objective

This thesis addresses this conflict by developing a data-driven framework for automated VVUQ of automatically generated, simulation-based DTs which have been learned from data. The focus lies on DMFS due to their practical relevance and dynamical, procedural nature. The research can further be specified by the following research questions (RQ):

- **RQ1:** How can automated validation and verification processes for DTs be efficiently implemented to maintain accuracy?
- **RQ2:** Which data-driven approaches are best suited to identify discrepancies between simulated behavior and real operational data in discrete material flow systems?
- **RQ3:** To what extent does the developed framework improve the quality and reliability of DTs compared to traditional VV methods?

This thesis proposes that object-centric event logs—commonly used to generate DTs in manufacturing—can also serve as the foundation for an automated, use-case-independent validation and verification framework. Such an approach would preserve the efficiency benefits of automated generation while ensuring that the resulting DTs meet necessary standards. A key aspect of this approach is the development and monitoring of generic, statistically grounded reference values, which must be quantifiable and have an underlying distribution. The framework will be evaluated using a case study from the discrete material flow

domain, providing empirical evidence of its effectiveness in improving model accuracy and efficiency.

1.4 Structure and Methodology

Structure

The thesis is organized into eight chapters. Chapter 2 establishes the theoretical foundation. It begins with broad, domain-specific concepts and progressively narrows the focus to the core topics of this thesis: Automated verification and validation (VVUQ) of simulation-based digital twins (SBDTs) in discrete material flow systems. Section 2.1 introduces material flow planning and simulation, outlining the key elements of production systems, such as processes, resources, and control mechanisms. It also defines key performance indicators (KPIs), essential for evaluating both real and simulated systems, providing the practical context in which DTs operate. Section 2.2 then transitions to the DT concepts. A framework for comparing DTs by **schwede2024learning<empty citation>** is presented. Special attention is given to data-driven DTs and their subset, automatically generated digital twins (AGDTs). The section concludes by contrasting AGDTs with classical simulation models, highlighting the challenges posed by automatically generated models. Building on this foundation, Section 2.3 presents the principles of process mining (PM) and event log analysis, focusing on object-centric event logs as a data basis for automated validation. This section demonstrates how PM acts as a bridge between real-world process data and model validation, thus enabling continuous verification of SBDTs. Section 2.4 narrows the focus further to VVUQ in the context of SBDTs, beginning with a historical overview of VVUQ methodologies. It then addresses the specific challenges posed by automatically generated models, such as data dependency and lack of transparency in model creation. The section introduces machine learning-based approaches for VVUQ, particularly classification methods for detecting model deviation. It also explores the current state of VVUQ in corporate practice, emphasizing the need for continuous and automated validation processes.

?? outlines the methodology for developing the proposed framework. It begins with a requirements analysis, deriving functional, technical, and data format requirements from theoretical findings. The chapter then elaborates on the data-based validation strategy, machine learning-based validation approach, metrics for model evaluation, and online validation with continuous monitoring.

?? presents the implementation of the framework, starting with the architecture and system setup, followed by detailed descriptions of event log processing, simulation integration, and the machine learning pipeline.

?? presents the case study results, evaluating the framework's effectiveness in improving DT quality. It describes the application scenario and data basis, the automatically generated DT, validation experiments, and result interpretation. It concludes with a comparison to manual validation methods.

?? discusses the implications of the results and provides recommendations for future research. It evaluates the framework in light of the research questions, examines the significance of verification in automatically generated DTs, addresses limitations, and explores implications for research and practice.

?? concludes the thesis by summarizing key findings and their implications. It addresses the research questions and hypotheses, discusses the significance of the results, acknowledges limitations, and provides recommendations for future work.

Methodology

The thesis follows a Design Science Research approach (DSR). This approach is characterized by the development of artifacts to solve practical problems (**hevner2004design**; **peffers2007design**). Artifacts in the sense of DSR are created objects or constructs which address the given problem and contribute to both theory and practice. The artifacts are evaluated in a real-world context to demonstrate their effectiveness. The thesis applies the cyclical DSR model, see figure 1.2.

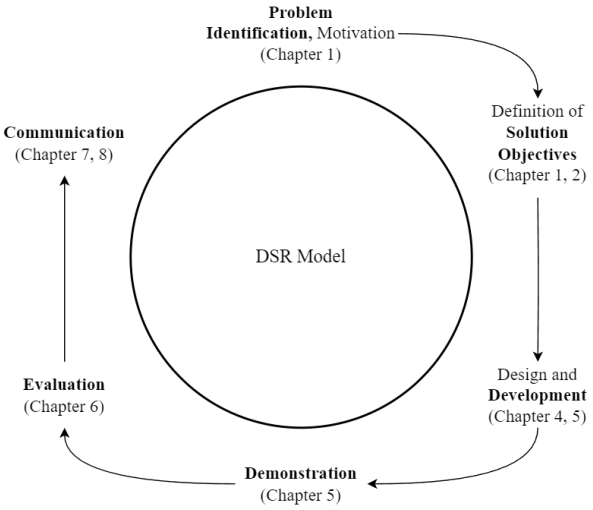

Figure 1.2: The cyclical design science research model. The model consists of six steps. The problem identification (1) refers to the research gap in automated VVUQ of SBDT. Defining the solution objectives (2) specifies the research gap by formulating questions and hypotheses based on the theoretical foundations. The design and development (3) phase includes the development of the framework. The demonstration (4) phase shows the application of the framework in a case study. The evaluation (5) phase assesses the effectiveness of the framework. The communication (6) phase concludes the research by presenting the results.

The research paradigm of the thesis is deductive-theory critical (**eberhard1987einfuhrung**). A conceptual VVUQ framework is developed based on existing theoretical foundations, while deriving new requirements through a requirements analysis. The framework is then applied in a case study to evaluate its effectiveness. The research is critical in that it aims to improve the efficiency and effectiveness of VVUQ for automatically generated DTs. Elements of empirical research are included through the case study and the data-driven approach.

Chapter 2

Theoretical Foundation

The following chapter provides a theoretical foundation for the research conducted in this thesis. It introduces the basic concepts of material flow planning and simulation, digital twins, process mining, and verification, validation, and uncertainty quantification (VVUQ). The relevance of these concepts in the context of simulation-based digital twins and their application in corporate practice will also be discussed.

2.1 Discrete Material Flow Systems and Simulation

This section begins with an introduction of the underlying concepts of Discrete Material Flow Systems (DMFS) and Simulation Based Digital Twins (SBDT).

2.1.1 Basic Concepts

Discrete material flow systems cannot be fully understood without first clarifying the principles of Discrete Event Simulation (DES) for Discrete Event Systems. In DES, a system changes its state through *events* that occur at specific, discrete time instances; it is assumed that no changes occur between two successive events. Consequently, the state of the system is completely defined by the values of its descriptive variables at each event occurrence (**varga2001discrete**). The time at which an event occurs is typically marked by a timestamp, and the scientific observation of such systems is conducted by analyzing the discrete *sequence* of events over time (**robinson2014simulation**).

Simulation, in this context, refers to the process of imitating the operation of a Discrete Event System over time—often through multiple event sequences. This imitation is captured in a model, and the core activities in a simulation

involve constructing and experimenting with this model. A high-quality simulation abstracts the essential features of the system, which requires the modeller to have a sound a priori understanding of what "essential" means in the given context. Although the model can later be refined, its quality is primarily measured by its ability to predict outcomes and offer a diverse range of scenarios (**maria1997introduction**).

In the context of DMFS, their simulation describes the imitation of material flow systems by breaking down continuous flows into discrete events. Such material flow systems can be characterized as "systems processing discrete objects (parts) that move at regular or irregular intervals along transportation routes or conveyor lines, comprising production and logistic systems" (**Arnold2006**; **schwede2024learning**). These systems form the backbone of material flow planning and control structures. The central idea of material flow planning and control is to ensure that material requirements—both in terms of quantity and timing—are met during transportation and storage across the various stages of the supply chain (**Gehr2007**). Importantly, the time horizon of interest spans from order placement up to delivery. To summarize, DMFS are often simulated using DES, which abstracts the continuous flow of materials into discrete events. The simulation is carried out using a model. The simulation and modeller are embedded in the context of material flow planning and control, which aims to ensure that material requirements are met across the supply chain. Successfully performed material flow planning and control induce high quality data for simulation and modelling purposes.

2.1.2 Comparing DMFS

Because the simulation of DMFS often involves (discrete) event simulation, events in DMFS need to be further differentiated to be comparable. (**Arnold2006**) propose to differentiate DMFS into static and dynamic components.

Static components describe the possible states of the system. Possible states can be the set of possible processes given a part or resource, for example. Dynamic components define the concrete material flow for a certain part or order. Static components include parts, resources and processes (**schwede2024learning**). Parts are transformed by processes using resources sometimes based on orders. Transformation can have an impact on physical properties of the parts (transformation model), spatial position (transition model), the quality of the parts (quality model) and takes time (time model) and uses resources (resource model). Resources have a capacity of handling parts in parallel (resource capacity model)

and processes have a predecessor-successors relationship (process model). Dynamic components are used to define the concrete dynamic material flow within the DMFS. There are four components: Order generation, order control, resource control and supply control. Order generation defines the load the system must process. Order control defines how parts are processed, sometimes referred to as routing rules (**mildeautomated**). Resource control defines how resources decide to handle processing requests, also sometimes referred to as priority rules. Supply control describes how supply parts are provided (**mildeautomated**; **schwede2024learning**). See the latter source for a more detailed description of the components.

2.1.3 Production Planning and Control

Successful companies use production planning and control frameworks to describe and optimize their DMFS. After establishing a theoretical foundation and simulation approaches for DMFS, this section thus focusses on Production Planning and Control (PPC) as a critical factor influencing the quality and quantity of data generated by Discrete Event Simulation. PPC is the structured approach to planning, scheduling, controlling and managing all aspects of the manufacturing process. It involves the coordination of resources, processes, and orders to meet production goals. PPC is essential for optimizing production processes, reducing costs, and improving quality. The main functions of PPC include production planning, production scheduling, and production control. Production planning involves determining the production capacity, production goals, and production processes. Production scheduling involves creating a detailed schedule for production activities. Production control involves monitoring and controlling production activities to ensure that production goals are met (**kiran2019production**). Scheduling is usually the last step performed before execution of the plan (**pinedo2012design**).

The integration of PPC with simulation models is crucial because it directly affects the data quality used in DES of DMFS. Effective PPC processes anticipate anomalies in the production cycle, allowing for adjustments that maintain system efficiency and reliability. If successful, these adjustments yield high-quality data that enhance the accuracy of simulation outcomes. (**kiran2019production**).

2.1.4 Relevant KPIs and Metrics

Up to this point, DES for SBDT of DMFS has been introduced, outlining the key factors that contribute to a robust simulation. A model differentiation framework proposed by **schwede2024learning** has been briefly presented to facilitate comparison of SBDT. Furthermore, the critical role of PPC in generating high-

quality data for simulation has been discussed. These discussions ignored up till now that, even when SBDT are integrated within well-functioning PPC processes, various SBDT models remain prone to errors and inherent trade-offs that must be addressed by the modeller (**Tao2018ijamt**).

The goal conflict of the modeller when developing SBDT can be described by the following conflict triangle (**robinson2014simulation**; **balci2012life**):

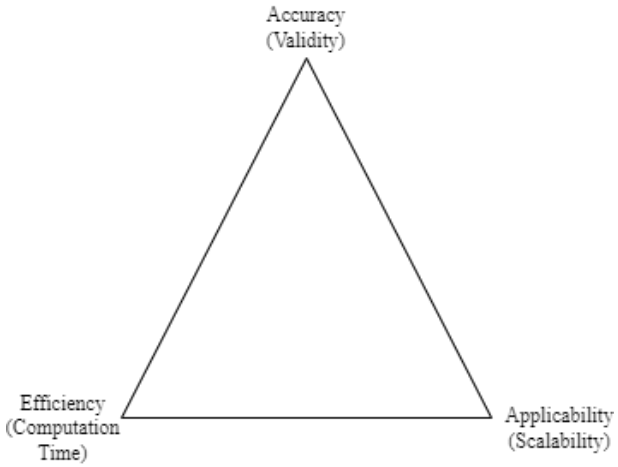

Figure 2.1: The goal conflict of the modeller when developing SBDT. Aiming for higher accuracy (validity) often leads to higher computational costs (efficiency) and reduced scalability (applicability). Reaching more efficiency often leads to reduced accuracy and scalability. Aiming for higher scalability often leads to reduced accuracy and efficiency.

Focusing one of the three dimensions—accuracy (validity), efficiency (computation time), and applicability (scalability)—often leads to trade-offs in the other two dimensions. Oftentimes the data itself is not sufficient to make a decision on which trade-off to make. Limited data points may hinder the modeller from reaching high validity. System architecture may block the system from reaching good scalability. Hardware limitations may hinder the modeller from reaching high efficiency. At other times, corporate management may have a preference for one of the dimensions.

One solution to balance and quantify these goals can be achieved by defining a set of KPIs. Some may already be available through PPC, some may be calculated from DES data or the DES itself. Optimally, the data warehouse provides relevant views (**cui2020manufacturing**). Because the SBDT in theory mirrors the DMFS, the KPIs gathered from PPC and the DES should yield identical values. Deviations between the KPIs of the SBDT and the DMFS may indicate errors in the SBDT or anomalies in the DFMS. The following KPIs are relevant for the evaluation of SBDT:

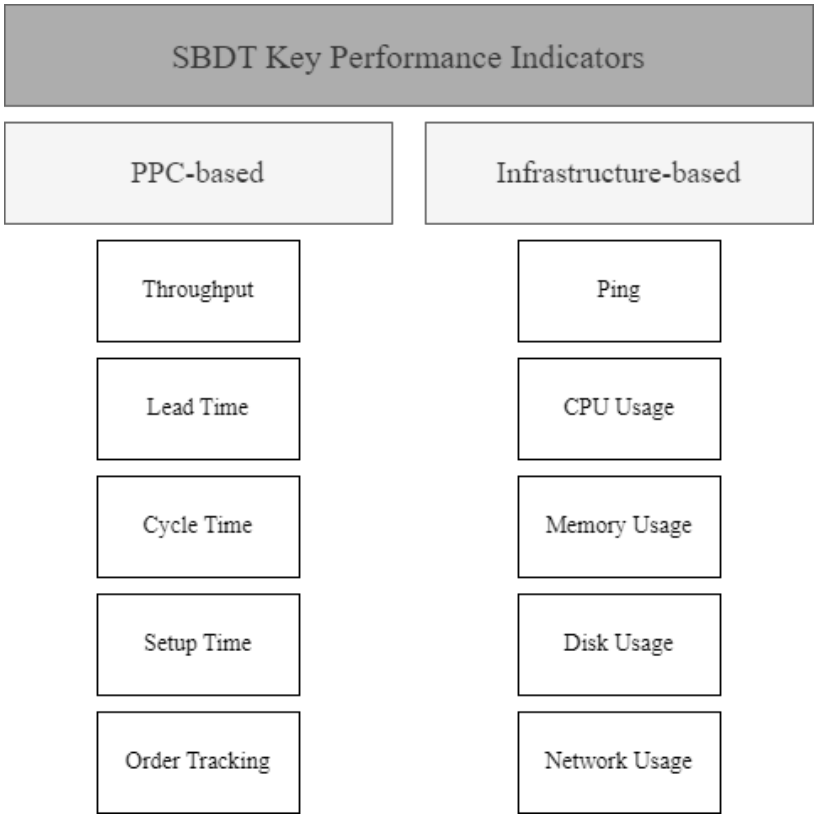

Figure 2.2: SBDT KPIs differentiated by PPC-based and Infrastructure-based indicators.

The PPC related KPIs may be provided by above mentioned data warehouse, because they are highly relevant in the context of production scheduling and -control. Throughput measures the number of produced parts at the last station in a specified period. It is an indicator for the productivity of the manufacturing sytem (**imseitif2019throughput**). Lead time is the cumulative time a part travels through the system from start to sink. It is an indicator for the efficiency of the manufacturing system (**pfeiffer2016manufacturing**). Cycle time measures the same amount like lead time but focusses only on the active production process, excluding transports and waiting times (**griffin1993metrics**). Setup time measures the time needed to prepare a machine for a new task. It is an indicator for the flexibility of the manufacturing system (**allahverdi2008significance**). In the given usecase, we aggregate the setup time for all setup processes. All KPIs presented so far can be calculated dynamically when new data has been sent. Later on, they may serve as an alert system for the modeller to detect deviations between the SBDT and the DMFS, see section 2.4.

The infrastructure related KPIs are derived by sensors from the executing system of the SBDT. Ping time measures the time needed to send a signal from one point to another. It is an indicator for the latency of the infrastructure (**wu2021digital**).

SBDT need to enforce real-time control over the physical entity. The latency thus needs to be as low as possible. In this scenario, one point (sender) is represented by the physical entity and its sensors. The receiving point runs the SBDT. It is advantageous to run the SBDT on Edge to minmize latency- and transmission costs (**li2018learning**). CPU-, memory-, disk- and network usage metrics are indicators for the load of the infrastructure. They are important to detect bottlenecks in the infrastructure (**li2018learning**). The first indicator is usually measured in percent of the maximum CPU capacity. The latter three indicators are usually measured in bytes or bits per second (**granelli2021evaluating**).

2.2 Digital Twin: Definition and Concepts

The latter section gave a short intrduction into DFMS, DES, its metrics and the corporate processes accompanying the SBDT. Now, we shed light on the DT itself. For a short introduction to the topic, see chapter 1.

Like introduced in the predecessing chapter, DT inherent the highest order of modelling fidelity compared to DM or DS. There are different definitions of DT present in the literature (**Negri2017promfg**; **zheng2019application**; **glaessgen2012digital**; **Demkovich2018def**; **boschert2016digital**; **grieves2014digital**; **kritzinger2018digital**; **Tao2018ijamt**; **zehnder2018representing**). Each of them highlights different aspects of the DT. This thesis uliziles the definition by (**grieves2014digital**) which highlights the conceptual elements of the twin and its lifecylce focus:

> The digital twin concept (...) contains three main parts: A) Physical products in real space, (B) virtual products in virtual space and (C) the two-way connections of data and information that tie the virtual and real products together. (**grieves2014digital**)

The physical product is the entity which will be modelled. The virtual product is the DT itself, but also its infrastructure, for example data services making the real-time data flow possible (**Tao2018ijamt**). The two-way connection is the data flow between the physical and the virtual product. The data flow is bidirectional. **zehnder2018representing** add that the data flow may contain meta data ”describing the data source and its context”. Also the connection protocol is of importance here (e.g. MQTT or REST). TCP may be the method of choice as it ensures that the packages arrive in the correct order and without errors (**li2018learning**).

2.2.1 Types of Digital Twins

Now that a unified understanding of DT has been established, this section focuses on how DT may be learned from different sources of information. The following list includes the most relevant types of DT with a focus on different kinds of information sources:

- Simulation-based DT (SBDT) (**Lugaresi2021aifac**; **martinez2018automatic**)
- Data-driven DT (DDDT) (**he2019data**; **Friederich2022**)
- Hybrid Digital Twins (HDT) (**luo2020hybrid**; **huang2023hybrid**)

SBDTs (**Lugaresi2021aifac**; **martinez2018automatic**; **boschert2016digital**) are based on DES. They utilize discrete event simulation (see section 2.1) to create a dynamic representation of the physical system (**schluse2016simulation**; **pantelides2013online**). To incorporate a SBDT into workflows and processes, suitable data structures must be in place beforehand (**boschert2016digital**). DES may improve the predictive capabilities of the model compared to manual twin creation. DES is able to model causal relationships between the events (**francis2021towards**). In contrast, the development of a realistic simulation model requires experts and time (**Charpentier2014**). If the simulation model fails to capture recent behaviour of the physical entity, a recalibration is mandatory (**Friederich2022**). To conclude, SBDTs are a step forward to speed up the creation and updating processes of DTs.

DDDT rely on the utilization of data to model the physical entity. The data may be gathered from sensors, data warehouses or other sources. The data is used to train a model which represents the physical entity. The model may be a neural network, a decision tree or another machine learning model. The model is then used to predict future states of the physical entity. The model may be updated with new data to increase its accuracy (**he2019data**; **Friederich2022**). For a more detailed description of DDDT including its up- and downsides, see subsection 2.2.2.

HDT combine different sources of information to create a more accurate model of the physical entity. The sources may be simulation models (see section 2.1), data-driven models (see subsection 2.2.2) or physics-based models. Physics-based models contain information about the physical properties and behaviors of the entity. They do not have to learn these characteristics from the data because this information is made available to the model *a priori* (**kapteyn2022data**; **aivaliotis2019methodology**). The simulation based models accompanying the

physics-based one obeys characteristics of SBDT, see above. The combination of different sources may make the HDT more robust and a faster learner. HDT unite the advantages of SBDT with the knowledge advantage physics based models have. Unfortunately, they also inherit the disadvatages of SBDT. Physics-based models may also involve heavy involve heavy computational costs and domain expertise (**kapteyn2022data**).

2.2.2 Data-Driven Digital Twins

While SBDTs and HDT possess not negligible computational costs and require domain expertise, DDDT are able to learn from data without the need for a hand-written simulation model. The DDDT *learns* the model. Learning in the context of DDDT is not trivial, several approaches have been proposed in the literature (**he2019data**; **Friederich2022**; **francis2021towards**). Often times Data Science methods come to work. The learning process may be supervised or unsupervised. Supervised learning uses labeled data to train the model (**cunningham2008supervised**). The label can symbolize different values of interest. Unsupervised learning uses unlabeled data to train the model (**barlow1989unsupervised**). Often times, the task at hand is to group the data into different categories. Data sources of interest may be process data, resource data or order data (**Biesinger2019**). The learning process may be online or offline. Offline learning uses the data *once* for training, validation and testing, while online learning continuously updates the model with new data to adapt to changes in the physical system. Online learning is thus able to capture new trends in the data and to foresee concept drift (**tsymbal2004problem**). DDDT have to be differentiated from data-driven simulation (**Charpentier2014**), which involves human invervention to create highly individual solutions for the physical entity. The key difference is that every characteristic has to be explicitly described in the model by the expert, there are no efforts to let an intelligent algorithm learn these by itself. DDDT may be able to update themselves to new trends in the data by online learning, termed *syncronization* (**reinhardt2019survey**). Latter has to be differentiated from *updating*, which is a manual process to take corrective action in the logic of the twin itself (**schwede2024learning**). An example for updating a DDDT may be the addition of a new feature to the model. An example for synchronization may be the adaption of the model to new trends in the data. The latter may be done by the model itself, the former has to be done by the modeller. DDDT thus rely less on domain expertise and manual model creation. A suitable model may be able to capture relevant trends in the data and to predict outcomes which describe most of the characteristics of the physical entity. (**francis2021towards**) propose

several key elements a DDDT must contain to be termed *data-driven*:

1. **Data Collection:** The relevant entities to be modelled have to be identified. This activity involves data gathering of the identified entities and ensuring a steady data stream to a database. The data may be gathered from sensors, data warehouses or other sources.

2. **Data Validation:** This step involves cleaning and preprocessing the data. The data may contain missing values, outliers or other errors. The data has to be cleaned and preprocessed to ensure a high quality of the model. Plausibility checks may be performed to ensure the data is correct.

3. **Knowledge Extraction:** After the data has been collected and cleaned, events have to be detected. **francis2021towards** utilize process mining terms in this context, such as evemt detection and process discovery. The main goal in this step is to find a common ground on which events are of interest. The thesis later dives deeper into Process Mining techniques applied here, see section 2.3.

4. **(Semi-)automatic Simulation Modeling:** The data is used to train a model. This step may use offline or online data as a stream. The model may be a neural network, a decision tree or another machine learning model. The model is then used to predict future states of the physical entity. The model may be updated with new data to increase its accuracy.

5. **Continuous Model Validation:** Interestingly,**francis2021towards** propose a continuous model validation. In the online learning case, they recommend to use the steady data stream to apply validation techniques continuously, see section 2.4 The validation may be performed by comparing the model predictions with the real data. If the model deviates from the real data, the model may be recalibrated.

DDDTs go one step further than SBDT and minimize the influence of the human in the loop (**francis2021towards**; **Friederich2022**). Faster model development and updating activities are the result. The third reason to automate DT endevours elaborated by **schwede2024learning**, increasing prediction quality, rises and falls with the data quality, thus the gathering and preprocessing efforts of the modeller. Extrinsic factors like the number of data points available also play into the equation. If the number of features is greater than the number of samples, the curse of dimensionality hinders a good modelling performance (**koppen2000curse**). DDDT should avoid biased or noisy predictions at all costs. The identification of *relevant* events poses the risk of introducing a selection bias,

rather a confirmation bias. The modeller may have the tendency to select events which confirm his or her hypothesis. Ramdom sampling may be a solution to this problem, but can destroy sequential information patterns in event sequences. Overall DDDT are a promising approach to model the physical entity. If the right balance between human involvement and automated learning is found, it may be an efficient solution (**francis2021towards**). Thinking one step ahead, employing data-based VVUQ approaches may also be a gamechanger. This topic will be discussed in Section **??**.

One last discipline, automatic simulation model generation (ASMG), is worth mentioning. ASMG has to be differentiated from DDDT by the effort to automatically generate models, thus DM and DS through DES. Automatic DT generation is not necessarily the goal. It aims to automate the model generation process and tries to eliminate the human in the loop, (**reinhardt2019survey**; **lechevalier2018methodology**). Automation is achieved by taking into account a diverse range of data sources, including Computer Aided Design data, PPS data, production manuals, process data and programming code, thus reaching a high data variability. The gathered data has to be processed online or offline as well through suitable frameworks or human intervention. Challenges lay in incomplete data (**bergmann2014automatische**), although the same problems of DDDT also apply here. If the gained data is not mined thoroughly, human intervention is needed again, mitigating automation efforts.

To conclude this section about DT, the thesis summarizes that there are different types of DT differentiated by their source of information retrieval. A lot of work has been done to make the DT creation, updating and prediction process more efficient. By the help of simulation, data and automated model generation, the DT may be created with less time and resources than manually.

2.3 Process Mining and Event Logs

After we introduced the corporate embedding of DTs and their types, the thesis now focusses on process mining (PM) and event logs. PM is a discipline which aims to extract knowledge from event logs. Event logs are the data basis for PM. The following section introduces the basic concepts of PM and event logs.

2.3.1 Core Concepts

PM is a discipline established 1999 which is interdisciplinary rooted in the field of Data Science and Process Science (**van2016data**). Data Science can be con-

sidered a process agnostic discipline (**van2016data**) while process science uses models not covering hidden trends in the data. The bridge between both approaches is PM. The goal of PM is to use event data to identify and extract process information (**vanderAalst2012**). This information is used to discover (process discovery), monitor (conformance checking) and improve processes (process enhancement) (**vanderAalst2012**) by using event logs. Such logs must contain a case ID, an activity name and a timestamp. Additional information like resource informations, order informations or other context information may be added to the log (**vanderAalst2012**). Such logs assume that the process can be captured fully and sequentially.

Table 2.1: A fragment of a manufacturing event log: Each line corresponds to an event. The case ID groups unique events which are identified by an event ID to one group, a trace. The timestamp refers to the time of event occurrence, while the activity describes the event. In this example, additional information like resource name and cost are given as well. Case 1, for example, consists of five events, involving a warehouse, two inspectors, and one machine.

Case id	Event id	Timestamp	Activity	Resource	Cost
1	101	10-01-2025:08.00	receive raw material	Warehouse A	500
	102	10-01-2025:08.30	initial quality check	Inspector Stefan	300
	103	10-01-2025:09.00	cutting process	Machine X	800
	104	10-01-2025:09.45	assembly	Worker Paul	600
	105	10-01-2025:10.30	final inspection	Inspector Eva	400
2	201	11-01-2025:07.45	receive raw material	Warehouse B	500
	202	11-01-2025:08.15	cutting process	Machine Y	800
	203	11-01-2025:09.00	welding	Robot Arm Z	700
	204	11-01-2025:09.45	quality assurance	Inspector David	400
3	301	12-01-2025:06.30	receive raw material	Warehouse C	500
	302	12-01-2025:07.00	initial quality check	Inspector Claudius	300
	303	12-01-2025:07.30	CNC machining	Machine W	900
	304	12-01-2025:08.15	painting	Worker Daniel	500
	305	12-01-2025:09.00	packaging	Worker Johannes	350

Table 2.1 illustrates the PM concepts. The case ID groups unique events which are identified by an event ID to one group, a trace. The timestamp refers to the time of event occurence, while the activity describes the event. In this example, additional informations like resource name and cost are given as well. Cases containing the same events identified by unique event IDs will have different case IDs (**van2016data**). Process discovery may try to produce a process model from the event log. The model may be a Petri net, a BPMN model or another process model. The challenge lies not in the recording of every trace present in the event log, rather in finding a generic representation of the most occuring traces. The process model must be generic enough to describe most traces, but specific enough to not get invalidated by future traces which may contain com-

pletely different events. Another major building block of this process model is accounting for trace concurrency. When several events may be identified to happen in parallel during the same time window, the model must recognize this. It can be spoken of a classical bias-variance tradeoff lend from data science. The process most contain the most frequent traces but has to filter out traces which contain anomalies. Such anomalies like longer time per event due to a fire alarm have to be accounted for. Conformance checking may compare a given process model against a given event log. They are specialized in detecting aforementioned anomalies. A key insight in conformance checking lies in two possible deviations from reality: The given model does not capture the real behaviour (A) or reality differs from the model (B). In the first case, the model is not working as intended. In the second case, the event log is corrupt. The third view on event logs, process enhancement, enables the modeller to use the generated process model to identify bottlenecks. Anomalies identified serve as a good starting point because they reveal errors in the process sequence. The given table offers costs associated to each event ID. This information may be used to create an event benchmark to further optimze the desired ”ideal” trace. The first goal of course is to ensure that no mistakes happen during a process.

More generally, PM empowers the modeller to perform VVUQ of an process model, event log or described trace. This concept is captured by the terms *Play-In*, *Play-Out* and *Replay* (**damm2001lscs**), see Figure 2.3.

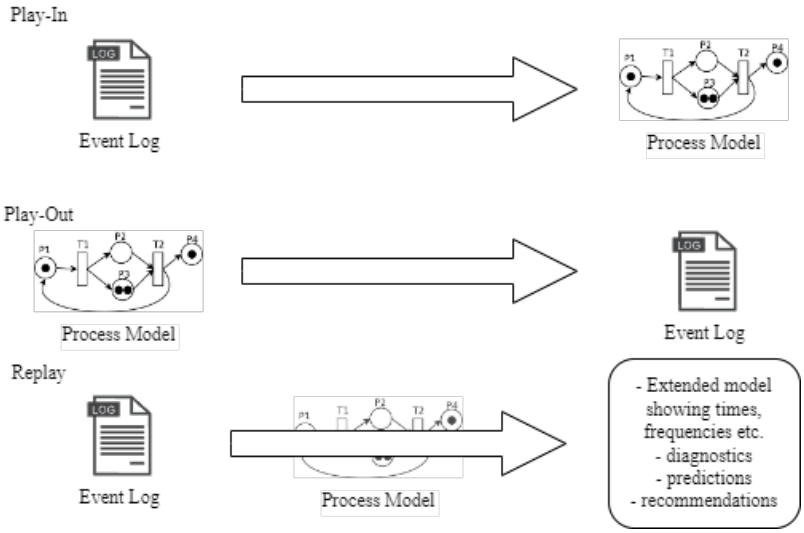

Figure 2.3: The Play-In, Play-Out and Replay concept in the context of process mining. The Play-In phase involves the creation of a process model from an event log. The Play-Out phase involves the creation of an event log from a process model. The Replay phase involves the modification of the process model thorough information gained from the event log.

Play-In refers to the creation of a process model out of an event log. Play-Out

may be called ”sampling” in data science as it generates traces out of the model. DES uses Play-Out to generate new exemplary behaviour, see section 2.1. Here, a process model may be used to play-out (simulate) several traces of the desired events and then use bagging techniques like averaging the durations per event to gain robust KPIs (subsection 2.1.4) and to reduce variance. A biased process model may still generate biased KPIs. Replay uses event log and process model together to check conformance, enriching the model with information captured in the log or to transform a descriptive process model into a prescriptive one (**van2016data**).

PM uses different formats to display process models like petri nets and BPMN models, among others (**vanderAalst2012**). The thesis at hand focusses on DDDT in the context of simulation. Such data-driven models often come as a black box, thus they can not be rendered by PM models.

2.3.2 Object-Centricness in Process Mining

The problem with traditional PM lies in its single dimensionality of perspective. For each process question, a new play-out has to be performed. Interactions between different objects are not captured (**van2023object**). One event may be related to different cases (convergence) or from the perspective of a case, there may be several equal activities within a case (divergence) (**van2019object**). Recently, object-centric event data (OCED) have been proposed as a new data basis for PM (**van2019object**). OCED logs (OCEL) are a generalization of classical event logs. Such traditional event logs use the case ID as a label to group and distringuish events. They assume that the process model describes the start and end of a single object. Each event refers to exactly one object (case) (**van2023object**). OCED overthrows these assumptions by assuming that events may relate to multiple objects. To account for this new logic, OCEL coins the terms events, objects, types, attributes and qualifier. Each object has exactly one object type. Several objects may have the same type. An object type can be a description of the function such as machine, customer, supplier or activity descriptions such as invoice, request. Objects are instances of these types. Events in particular have an event type, termed activity. The same non-uniqueness applies here —many events can have the same type like ”processing complaint”, ”cooking coffee”. Each event is described by exactly one type. OCED assumes event atomicity; each event is indivisible. Each event has one timestamp. Compared to traditional PM, events may relate to multiple objects through a qualifier (E2O). Such a qualifier may be, considering the event ”printing label” and object ”printing station”, the label to be printed. Objects may be related to multiple ob-

jects (O2O). O2O relationships are frozen (static) and are assumed to not change. The O2O relationship may be used to describe the order of producing a product. For example, the O2O relation ”main pcb to gyroscope” may say that the main pcb has to be produced before the gyroscope. Another O2O relation can be an order, the connection between the customer object and the ordered product. It is worth mentioning that objects can also be related indirectly together through two E2O relations: The E2O relation ”producing” may connect the event ”machine 1 produces” with the object ”gyroscope”. The E2O relation ”check” may connect the event ”machine 1 checks” with the object ”main pcb”, thus connecting the two objects ”gyroscope” and ”main pcb” indirectly via two events (**van2019object**). E2O relations are dynamic. They may change over time, involving different objects. In the given example, ”main pcb” would be checked with ”display” instead of ”gyroscope”.

Events and objects have attributes (keys), possessing values. Event attribute values refer to exactly one event and one event attribute, they are not shared. For example, the cost for one event may have the value $10€$. The same logic applies to objects as well, one event attribuite value refers to exactly one object and one object attribute. Because several events may have the same type and several objects may have the same type as well, each event- or object type may refer to any number of event or object attributes. They open interpretative possibilities for the modeller by considering them as expected attributes for the event or object type. The given example may be the event type ”producing” which may have the event attribute ”duration” and ”cost”. The object type ”gyroscope” may have the object attribute ”weight” and ”size”. One may average the different object attribute values of one type cluster to generate object type KPIs. A key specificity of object attribute values lies in the fact that they have a timestamp. Event attribute values do not. Latter information would be redundant because events to already have a timestamp, see above. Event attribute values may have cardinality one per event. N event attributes have n values. This does not apply to object attribute values. They may have multiple values because of their nature to change over time (**van2023object**). This is why each object attribute value has a timestamp. The attribute value is in conclusion unique for a given object during a given time given one attribute. The given example may be the object attribute ”weight” of the object ”main pcb” which may change over time. The object attribute value ”weight” may have the value $100g$ at time t_1 and $120g$ at time t_2.

OCEL thus extends traditional PM by accommodating the multi-object nature of complex processes. Unlike classical event logs—which restrict events to

a single case—OCEL captures dynamic interactions among multiple objects via E2O relations and static inter-object dependencies through O2O relations. This enriched framework enables a more comprehensive representation of real-world processes, where events may concurrently affect several objects. Timestamped object attribution values enable the modeller to perform temporal analysis and KPI derivation (subsection 2.1.4). Overall, OCED provides a more detailed, semantically rich representation of complex, interconnected processes. (**van2023object**).

2.3.3 Process Mining as Enabling Technology

Recall that PM uses event logs to develop process models or to enhance existing ones, so these logs may serve as a foundation for VVUQ of models in general. Live twin data often can be exported to the event log format. Several standardizations have been proposed, with the IEEE XES standard being the most widely used (**van2016data**). The XES standard defines a common format for event logs, enabling the exchange of event data between different software frameworks. The idea lies in exporting twin decisions or live data as event logs and then use PM tools to perform VVUQ. Replay may be used on SBDT simulated traces in comparison with actual event data to reveal mismatches in sequences or timing. For example, the SBDT could have predicted a circual process. Replay can be applied to further analyze this bottleneck. Play-Out can empower the modeller to sample a big amount of exemplary traces to gain KPIs. OCED object attribute values may offer even more insights. PPC systems (subsection 2.1.3) often times deliver even more data to enrich the event log so that VVUQ can be performed easier. If event log extraction out of the SBDT is performed online, VVUQ can be performed on the fly.

2.4 VVUQ in the Context of Simulation-Based Digital Twins

The previous sections introduced the concepts of DES, PPC, relevant KPIs, DT, PM and OCED. This section now focusses on VVUQ in the context of SBDT. The thesis at hand uses the term VVUQ to describe the process of verifying, validating and quantifying the uncertainty of a SBDT. Verification and validation has a long history in manufacturing and DES (**Bitencourt2023**). (**sel2025survey**) add uncertainty quantification as a main interest. Their framework is applied in the medical domain, but they mention reasonable arguments regarding efficiency and safety of SBDT in general. Thus, the thesis considers VVUQ instead

of merely verification and validation efforts. The following section introduces the basic concepts of VVUQ and its relevance for SBDT.

2.4.1 Development process of VVUQ Concepts

With the uprising of simulation models in the early 1950s (**evans1967simulation**), the need for VVUQ arose unknowingly to the modellers. The usability of such simulations was deemed high as long as the results were promising, increasing trust in the technology (**durst2017historical**). Blind trust does not validate models. Contrarily, if the results more or less were satisfactory, the model was considered validated (**bonani2003physics**).

The first effort to define and perform verification was performed by (**machlup1955problem**) defining verification as ”including the correctness of mathematical and logical arguments, the applicability of formulas and equations (...), the reliability and exactness of observations, the reproducability of experiments, the explanatory or predictive value of generalizations.”. (**naylor1967verification**) further refined this definitions by introducing the idea of ”goodness of fit”. Latter describes the capability of the model to correctly reflect the modelled system. During the 1970s, researchers like Schlesinger (**schlesinger1979terminology**) defined validation as achieving a ”satisfactory range of accuracy consistent with the application”, while Ignall argued for validation against simulations rather than analytical models (**ignall1978using**). Sargent’s work from 1979 to 1984 proposed methods like user collaboration and independent VV, detailing techniques such as sensitivity analysis and Turing tests (**Sargent2010wsc**). Balci developed a taxonomy and emphasized continuous VV, reflecting the need for ongoing assessment (**balci2012life**).

By 2004, modern VV emerged. Considering model fidelity of today’s approaches, (**Oberkampf2004amr**) introduced widely recognized definitions of VV:

> *Verification* is the process of determining that a model implementation accurately represents the developer’s conceptual description of the model and the solution to the model.
>
> *Validation* is the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model. (**Oberkampf2004amr**)

Verification thus concerns itself with the correctness of the given model (**Sargent2010wsc**), while validation evaluates the quality of explanations quantitavely (**Oberkampf2004amr**). **iso2017systems** span the concept of validation to computational models, where

valid models correctly reflect the users intented functionality. PPC may provide the modeller with relevant KPIs to assess both.

Uncertainty quantification (UQ) is a relatively new field in VV. It aims to quantify the uncertainty of the model and its predictions through the whole lifecylce of the model, including training, inference and prediction (**sel2025survey**). Uncertainty quantification empowers the modeller to define confidence intervals to correctly emphasize the stochastic nature of the model predictions (**volodina2021importance**). This is crucial for SBDT, where real-time decisions rely on accurate representations, as (**francis2021towards**) highlights the need for continuous updates. UQ differentiates *aleatory* uncertainty, which is inherent to the data, and *epistemic* uncertainty, which is due to lack of knowledge (**sel2025survey**). Aleatory uncertainty may arise due to sensor erros in the physical entity, generating noise. Epistemic uncertainty may arise due to lack of data or knowledge about the physical entity (**thelen2023comprehensive**). Epistemic uncertainty can be reduced. (**abdoune2022handling**) provide a comprehensive overview of UQ challenges and potential solutions in the context of SBDT.

For SBDT, VVUQ is not a one-time activity but a continuous process, ensuring digital twins mirror physical systems accurately. This is vital for industries like manufacturing, where decisions based on digital twins have significant implications section 1.2. The historical development of VVUQ, from early verification to integrated UQ, reflects the growing complexity of simulation models. As SBDT become central to decision-making, robust VVUQ practices ensure reliability, linking back to foundational concepts introduced earlier, see subsection 2.2.1. The concepts of VVUQ are embedded in the context of PM and OCED, which may further assist described endevours. Especially for UQ, PM offers a rich data source to quantify uncertainty and validate models by providing a *degree of fit* of the given process model.

2.4.2 VVUQ for Automatically Generated Models

AMG stem largely stem from the DES discipline (**??**), termed ASMG (**mildeautomated**; **Charpentier2014**). They may use data-driven techniques which reduce the manual effort to create and update themselves. These models adapt quickly to changing conditions, making them valuable for dynamic environments like manufacturing. However, ensuring their reliability requires specific VVUQ. Because ASMG models often are black boxes, traditional VVUQ methods may not be applicable. The challenge lies in understanding the model's internal logic and ensuring it accurately reflects the physical system. If sophisticated data driven

methods have been applied, several *key challenges* arise, hindering successful VVUQ:

- **Model Opacity:** ASMG models often use sophisticated machine learning algorithms like neural networks. Such black-box models are difficult to interpret, making it hard to understand their internal logic. This opacity may hinder VVUQ, as the modeller cannot easily identify errors or biases.
- **Data Dependency:** Data-driven models rely on high-quality data. If the data is biased or noisy, the model's predictions may be inaccurate.
- **Dynamic Model Adaptation:** Models that continuously learn and adapt, such as those employing online learning, require ongoing validation to ensure they remain valid as new data is ingested. This dynamic nature introduces the risk of concept drift (**lu2018learning**), where the underlying process changes over time, potentially degrading model performance.
- **Quantifying Uncertainty:** Model predictions are stochastic in nature. For applications where precision is crucial, such as in manufacturing, uncertainty needs to be quantified.

To assess these challenges, the thesis defines the following *key requirements* for VVUQ of automatically generated models, especially SBDT:

- **Model Interpretability:** Ensuring the model's internal logic is transparent and understandable, enabling the modeller to identify errors or biases. Developing and employing techniques to make the decision-making processes of automatically generated models more transparent is crucial for VVUQ. Explainable AI methods, such as SHAP (SHapley Additive exPlanations, (**lundberg2017unified**)) or LIME (Local Interpretable Model-agnostic Explanations, (**ribeiro2016shouldI**)), can help shedding light on model behavior, facilitating verification efforts. This is particularly important for black-box models like deep neural networks, where understanding the reasoning behind predictions is challenging. Often times, it may be easier to use white-box models like decision trees or linear regression models, which are interpretable by design. Such models are often times labeled transparent models. If black-box models are instead chosen, so called *surrogate models* may be used to approximate the black-box model. The surrogate model is a white-box model, which is easier to interpret. This approach is called *post-hoc* explanation (**fischer2024demystifying**). SBDT generating explanations by themselves are the ultimate goal.

- **Upholding Data Quality:** Implementing procedures to ensure that the data used for model generation and validation is accurate, complete, and representative of the operational environment is vital. This includes data cleaning, preprocessing, and plausibility checks to identify and mitigate issues like missing values, outliers, or biases. For instance, in manufacturing digital twins, sensor data must be validated for accuracy and consistency to ensure reliable model outputs (**rodriguez2023updating**).

- **Validatable Algorithms:** Besides ensuring model interpretability, the algorithms used for model generation must be validatable. Security risks are a huge concern for companies employing SBDT into their processes (**alcaraz2022digital**). The algorithms used have to be secure, hardened against attacks. This is especially important for online learning algorithms, which may be vulnerable to adversarial attacks (**balta2023digital**). Manipulated data may lead to incorrect model predictions, causing severe consequences in safety-critical applications like autonomous driving or medical diagnosis. Ensuring the robustness of the algorithms is crucial for reliable VVUQ.

- **Continuous Validation:** VVUQ processes have to work in real-time to ensure the model remains valid as new data is ingested. This requires continuous monitoring and validation of the model's predictions against real-world data. Techniques like online validation (**francis2021towards**) can help ensure the model's accuracy and reliability over time. This is particularly important for SBDT, which are designed to adapt to changing conditions and provide real-time insights. Continuous validation ensures the model remains reliable and trustworthy, even as the underlying process evolves.

- **Integration:** VVUQ processes have to be integrated into existing model- and PPC infrastructure to be able to perform VVUQ on the fly. This requires close collaboration between data scientists, domain experts, and IT specialists to ensure the seamless integration of VVUQ processes into the model lifecycle.

- **Scalability:** VVUQ processes have to be scalable as the underlying model or data evolves over time. This requires the development of scalable VVUQ techniques that can handle large volumes of data and complex models. Of course, the infrastructure must be able to handle the increased computational load.

As noted by **francis2021towards**, the lifecycle SBDT has implications for its VVUQ as well: VVUQ must accompany the SBDT in all phases, from conceptualization to deployment and operation. The key requirements outlined above provide a foundation for developing robust VVUQ processes for automatically generated models, ensuring their reliability and accuracy in real-world applications.

2.4.3 Traditional versus Machine Learning-Based Approaches

VVUQ can be performed using traditional methods or more sophisticated approaches. Traditional verification techniques may include code inspection, unit testing or debugging (**maniaci2018verification**). If closed solutions to simulated models are available, they may be used to validate the simulation. Traditional validation techniques involve comparisons between model predictions and real-world data, such as statistical tests or sensitivity analysis. In the context of manufacturing, simple experiments or historical data can be used. The consultation of experts is another possibility (**shao2023credibility**).

Machine Learning-Based VVUQ

Sophisticated approaches may include the use of machine learning (ML) techniques to enhance VVUQ processes. ML can be used to identify patterns in data, detect anomalies, and improve model predictions. In addition to supervised and unsupervised learning shortly introduced in subsection 2.2.2, semi-supervised learning and reinforcement learning are two other approaches. Supervised learning may be employed where labels regarding the validity or non-validity of an OCEL (see subsection 2.3.2) are given. Unsupervised learning may provide such labels as learned categories if they are missing or may assist with finding common patterns in the data. Unsupervised techniques are context-agnostic and assume no patterns in the data, see **hastie2009unsupervised** for the reverse problem of encoding a priori information in unsupervised algorithms. Semi-supervised learning combines labeled and unlabeled data to improve model performance. It somehow forms a middle ground where lots of unlabelled data is available and is then grouped by the algorithm. The scarce data is then used in the holdout set to perform testing (**learning2006semi**). Reinforcement learning is not commonly applied in VVUQ of SBDT. ML techniques are often referred to as ”oracles” when used for VVUQ because of their key challenge of opacity subsection 2.4.2.

Challenges in Data Preparation and Feature Selection

Before discussing the application of ML techniques in VVUQ, several problems hindering successful application of ML in VVUQ are identified. The first problem is data quality (**wu2025uncertainty**). Data quality may degrade the performance and reliability of ML-based VVUQ approaches. Firstly, *missing values* in the dataset can hinder the training process and potentially introduce biases. To account for this, imputation strategies or simply removing defect rows may be a solution (**gudivada2017data**). The modeller has to keep in mind that this may corrupt the VVUQ process. Secondly *outliers*, which are data points that deviate significantly from the rest, create a challenge as they might represent real trends in the data (and thus containing valuable edge cases for VVUQ) or simply be erroneous entries, requiring care of the modeller. Thirdly *noise* can overblend the underlying patterns that ML algorithms aim to learn, reducing the accuracy and generalization ability of the VVUQ models (**liu2020noise**). Such noise may be random, for example by measuring environmental influence, or systematic, for example emitted by erroneous sensors. The first kind of noise can be reduced by smooting or filtering.

Inconsistent data is another mistake to avoid in data gathering. It may arise from deviations in formats, units or representations of the same information. It can lead to confusion for ML algorithms and require standardization and cleaning (**mahanthappa2021data**). Duplicate data entries may also generate false trends in the data. Imbalance introduced by duplicates or measurement erros can bias the model in predicting only the majority class when a classification problem is at hand. Beyond data qualitiy, the modeller has to make sure that the data is representative and bias-free. The training data used for VVUQ models must accurately reflect the operational environment. Biases can arise through human intervention or environmental influence (**liu2020noise**). Finally, the way data is split into training, testing, and validation sets is important for avoiding overfitting. Here, the model memorizes the training data and fails to generalize to unseen data.

After ensuring data quality, the modeller has to consider suitable features for the given problem. Features are columns in the dataset describing characteristics of the data points (rows) through values. OCEL provide a relatively strict feature set in which the modeller has to operate. This has the advantage that several of the upper challenges may be eliminated because the data can be exported through a standardized interface. Successful VVUQ methods may require additional features nontheless. Only the most relevant features may be selected

(**geron2022hands**). Feature engineering describes the endevour of creating new features through combining existing features or through a new data gathering process. The incorporation of features in model training is coined feature selection. In the given use case, an adaptive feature a feature selection based on twin components may be useful, see subsection 2.1.2.

Classification Methods for the Detection of Model Deviations

Extensive work has been done on the topic of ML-based deviation detection. Such deviations are called ”anomalies” and the process is termed ”anomaly detection” (**kharitonov2022comparative**). Anomalies are data points that deviate significantly from the majority of the data. They can be classified into three categories: point anomalies, contextual anomalies and collective anomalies (**chandola2009anomaly**). Point anomalies are single data points that differ significantly from the rest of the dataset. Contextual anomalies are data points that are normal in one context but anomalous in another. They appear less often than point anomalies and have less statistical weight. Collective anomalies are groups of data points that are anomalous when considered together but may not be anomalous individually. Anomalies are of special interest in the context of VVUQ, as they can indicate potential issues with the model or the data, see **??**.

Several algorithms exist to detect anomalies, including statistical methods, clustering-based methods, and supervised learning methods. If the data is sufficiently labelled and preprocessed, supervised methods are promising. Algorithms such as Support Vector Machines (SVM), Neural Networks (NN), Decision Trees, and Random Forests can learn from this labeled data to classify new model outputs or behaviors as either normal or deviative. However, a common challenge in anomaly detection scenarios is the issue of class imbalance, where instances of normal behavior are more common than the occurrences of deviations. This imbalance needs to be carefully addressed during model training to prevent the classifier from being biased towards the majority class. Clustering-based unsupervised methods detect anomalies through grouping similar data points together. Algorithms like K-Means, DBSCAN, and Hierarchical Clustering can be used to identify clusters of normal behavior, with anomalies being those points that do not belong to any cluster. They measure the distance between data points to cull anomalous points as reaching far outside of a group of common data points. Often times the assigned groups can be plotted, yielding further insights in the nature of the detected anomalies. Statistical methods, on the other hand, rely on the assumption that the data follows a certain distribution. They identify anomalies based on statistical properties such as mean, variance, and standard deviation

(**chandola2009anomaly**).

Performance Metrics

Another important aspects of assessing model quality is through metrics. The following section introduces the most common metrics used in VVUQ of SBDT. The introduction is supported by an exemplary OCEL:

Table 2.2: A fragment of a manufacturing OCEL with type annotations. Each row represents an event with associated process execution details. The columns are annotated as follows: `process_execution_id` (int), `order_id` (int), `start_time` (datetime), `end_time` (datetime), `part_id` (int), `process_type` (int), `process_id` (int), `resource_id` (int), and `is_valid` (bool).

process_execution_id	order_id	start_time	end_time	part_id	process_type	process_id	resource_id	is_valid
0	2529	2020-04-22 14:21:07	2020-04-22 14:21:31	-1	0	0	0	True
1	2529	2020-04-22 14:23:35	2020-04-22 14:23:58	-1	1	1	1	True
2	2529	2020-04-22 14:21:09	2020-04-22 14:21:33	-1	0	0	0	True
3	2529	2020-04-22 14:23:36	2020-04-22 14:23:58	-1	1	1	1	True
4	2529	2020-04-22 14:21:11	2020-04-22 14:21:35	-1	0	0	0	True
5	2529	2020-04-22 14:23:36	2020-04-22 14:23:58	-1	1	1	1	True
6	2529	2020-04-22 14:21:13	2020-04-22 14:21:37	-1	0	0	0	True
7	2529	2020-04-22 14:23:36	2020-04-22 14:23:58	-1	1	1	1	True
8	2529	2020-04-22 14:21:15	2020-04-22 14:21:39	-1	0	0	0	True
9	2529	2020-04-22 14:23:37	2020-04-22 14:23:58	-1	1	1	1	True
10	2529	2020-04-22 14:21:18	2020-04-22 14:21:41	-1	0	0	0	True
11	2529	2020-04-22 14:23:37	2020-04-22 14:23:57	-1	1	1	1	False
12	2529	2020-04-22 14:21:20	2020-04-22 14:21:44	-1	0	0	0	False
13	2529	2020-04-22 14:23:37	2020-04-22 14:23:57	-1	1	1	1	False
14	2529	2020-04-22 14:21:22	2020-04-22 14:21:46	-1	0	0	0	False
15	2529	2020-04-22 14:23:38	2020-04-22 14:23:57	-1	1	1	1	False

Accuracy

Accuracy is a fundamental metric that quantifies the overall correctness of the classification model. It represents the ratio of correctly classified instances to the total number of instances in the dataset [Connolly2023, Grandini2020].

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

where TP denotes the count of true positives, TN represents true negatives, FP signifies false positives, and FN indicates false negatives.

This metric offers a high-level overview of the model's performance by indicating the proportion of predictions that align with the actual classes. While its simplicity makes it easily interpretable, accuracy can be a misleading metric when dealing with imbalanced datasets [Google, MDPI]. In manufacturing, where the occurrence of invalid processes might be significantly lower than valid ones, a high accuracy score could be achieved by a model that predominantly predicts the majority class, failing to effectively identify the critical minority class

of invalid processes [Google]. Therefore, relying solely on accuracy might not provide a complete or accurate assessment of the model's utility in identifying process anomalies.

Precision

Precision, also known as the positive predictive value, focuses on the quality of the positive predictions made by the model. It measures the proportion of instances that the model predicted as positive which were indeed positive.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.2}$$

A high precision score signifies that when the model predicts a manufacturing process as 'invalid' (assuming 'invalid' is the positive class), it is highly likely to be genuinely invalid. This is particularly important in manufacturing quality control to minimize the occurrence of false alarms, which can lead to unnecessary interruptions in the production line and increased operational costs [Google]. A model with high precision ensures that interventions based on its predictions are more likely to be warranted, thereby enhancing the efficiency of the quality assurance process.

Sensitivity

Recall, also referred to as sensitivity or the true positive rate (TPR), assesses the model's ability to identify all the actual positive instances within the dataset. It measures the proportion of actual positive instances that were correctly classified as positive by the model [Google, Iguazio].

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.3}$$

In the context of manufacturing, a high recall for the 'invalid' class is crucial for ensuring that the model effectively detects the majority of the truly invalid processes. Failing to identify an invalid process (a false negative) can have significant consequences, potentially leading to the production of defective goods that may incur costs related to rework, scrap, or customer dissatisfaction [Google]. Therefore, a model with high recall minimizes the risk of overlooking critical quality issues in the manufacturing process.

F1-Score

The F1-score provides a balanced measure of the classification model's performance by calculating the harmonic mean of precision and recall. This metric is particularly useful when dealing with datasets that exhibit an imbalance in the class distribution.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN} \tag{2.4}$$

By considering both the precision (the accuracy of positive predictions) and the recall (the ability to find all positive instances), the F1-score offers a single metric that summarizes the trade-off between these two important aspects of a classifier's performance. In the realm of manufacturing quality prediction, where achieving a balance between accurately identifying invalid processes and minimizing false alarms is often a key objective, the F1-score serves as a valuable tool for assessing the overall effectiveness of the ResNet Bi-LSTM Multihead attention network. It is especially beneficial when the costs associated with false positives and false negatives are relatively similar, or when a general measure of good performance across both precision and recall is desired.

Confusion Matrix

A confusion matrix is a specific type of contingency table that provides a detailed breakdown of the performance of a classification model by displaying the counts of true positives, true negatives, false positives, and false negatives. For a binary classification problem, such as predicting whether a manufacturing process is valid or not, the confusion matrix typically takes the form of a 2x2 table [PulmonChronicles, Wikipedia].

Table 2.3: Confusion Matrix for Binary Classification

Actual Class	**Predicted Class**	
	Positive (True)	Negative (False)
Positive (True)	True Positive (TP)	False Negative (FN)
Negative (False)	False Positive (FP)	True Negative (TN)

This matrix offers a granular view of the model's predictive behavior, allowing for a thorough analysis of the different types of errors it makes. True positives represent the cases where the model correctly predicted a positive outcome (e.g., an invalid process was correctly identified). True negatives indicate instances where the model correctly predicted a negative outcome (e.g., a valid process was

correctly identified). False positives occur when the model incorrectly predicts a positive outcome for a negative instance (e.g., a valid process was wrongly flagged as invalid). Conversely, false negatives arise when the model incorrectly predicts a negative outcome for a positive instance (e.g., an invalid process was missed and classified as valid) [Wikipedia]. Analyzing the confusion matrix for the 'is_valid' prediction in the context of manufacturing can reveal crucial information about the model's tendencies, such as whether it is more prone to generating false alarms or to missing actual defects, which has direct implications for the design and implementation of quality control strategies.

Using the provided data, and the hypothetical predictions:

```
Actual is_valid:
Predicted is_valid:
```

Assuming 'True' is the positive class, the confusion matrix would be: Here,

Table 2.4: Example Confusion Matrix for Provided Data

Actual Class	**Predicted Class**	
	True	False
True	8	2
False	1	1

$TP = 8, FN = 2, FP = 1, TN = 1.$

Classification Report

A classification report consolidates the performance evaluation of a classification model by presenting a summary of key metrics, including precision, recall, F1-score, and support, for each of the classes in the problem. The support indicates the number of actual occurrences of each class within the dataset [Kohli].

Example (based on the hypothetical predictions from the confusion matrix example):

	precision	recall	f1-score	support
False	0.50	0.50	0.50	2
True	0.89	0.80	0.84	10
accuracy			0.75	12
macro avg	0.69	0.65	0.67	12

```
weighted avg       0.82      0.75      0.78        12
```

This report offers a clear and concise overview of the model's effectiveness in classifying instances of each class. By providing class-specific precision, recall, and F1-scores, it allows for a more nuanced understanding of the model's performance, especially in scenarios where the classes are imbalanced or where the importance of accurate classification differs across classes [Kohli]. In the context of manufacturing process validity prediction, the classification report would detail how well the ResNet Bi-LSTM Multihead attention network performs in identifying both valid and invalid processes, providing valuable insights into its practical utility.

While the primary focus based on the example data is on classification, the ResNet Bi-LSTM Multihead attention network might also be employed for regression tasks in manufacturing, such as predicting continuous variables like throughput times or resource utilization levels. In such scenarios, the following regression metrics are crucial for evaluating the model's predictive accuracy.

Mean Squared Error (MSE)

Mean Squared Error (MSE) quantifies the average of the squared differences between the values predicted by the model and the actual observed values.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.5}$$

where y_i represents the actual value of the target variable for the i-th instance, $\hat{y}_i$ is the corresponding predicted value, and n is the total number of data points in the dataset.

MSE is a widely used metric that provides a measure of the overall prediction error. The squaring of the differences means that larger errors contribute more significantly to the final MSE value, making it particularly sensitive to outliers in the predictions. In the context of manufacturing, if the model were predicting throughput time, a high MSE would indicate that, on average, the squared difference between the predicted and actual throughput times is large, suggesting a lower accuracy in the model's predictions.

Mean Absolute Error (MAE)

Mean Absolute Error (MAE) measures the average of the absolute differences between the predicted and actual values.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.6}$$

MAE offers a more direct and interpretable measure of the average magnitude of the errors in the model's predictions. Unlike MSE, MAE treats all errors equally, without giving disproportionate weight to larger errors. In manufacturing applications, such as predicting resource utilization, MAE would represent the average absolute percentage point difference between the model's predictions and the actual utilization rates, providing a clear indication of the typical size of the prediction errors.

Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) calculates the average percentage error between the predicted and actual values.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\% \tag{2.7}$$

MAPE is particularly useful when the scale of the target variable varies, as it provides an error measure in relative terms. The percentage error is often easier to understand and communicate than absolute errors, especially in a business or operational context. For example, if the ResNet Bi-LSTM Multihead attention network were predicting the percentage of defective parts produced, MAPE would indicate the average percentage by which the model's predictions deviate from the actual defect rates, offering a valuable perspective on the model's performance in predicting proportional outcomes.

Performance Evaluation using ROC Curves and AUC

For binary classification tasks, such as the prediction of 'is_valid' status in manufacturing processes, Receiver Operating Characteristic (ROC) curves and the Area Under the Curve (AUC) provide a comprehensive evaluation of the model's performance.

An ROC curve is a graphical representation that plots the true positive rate (TPR or recall) against the false positive rate (FPR) at various classification thresholds. By examining the curve, one can visualize the trade-off between the model's sensitivity (its ability to correctly identify positive instances) and its specificity (its ability to correctly identify negative instances) across different decision points.

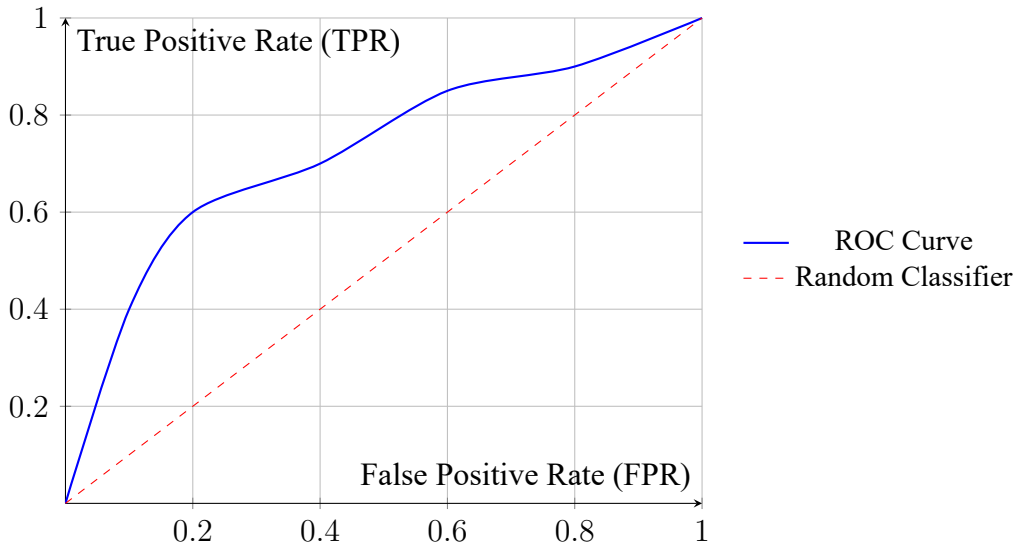

Figure 2.4: Example ROC Curve

The AUC metric quantifies the overall ability of the classifier to distinguish between the two classes. It represents the area under the ROC curve, with values ranging from 0 to 1. An AUC of 1 indicates a perfect classifier, while an AUC of 0.5 suggests a performance no better than random guessing. In the context of predicting 'is_valid' in manufacturing, a higher AUC for the ResNet Bi-LSTM Multihead attention network would signify that the model is better at distinguishing between valid and invalid processes, regardless of the chosen classification threshold. This is particularly valuable when the class distribution is imbalanced, as ROC and AUC provide a more robust evaluation compared to metrics like accuracy that can be skewed by the majority class.

Bidirectional LSTM Networks for Sequence-Based Anomaly Detection

The basic structure of RNNs and LSTM networks How bidirectional LSTMs work and their advantages Why BiLSTMs are particularly suited for sequence-based anomaly detection Brief mention of their applications in time series analysis

Anomaly detection and VVUQ share common goals of identifying deviations from expected behavior and ensuring the reliability of models. The two fields can benefit from each other, as VVUQ can provide a framework for evaluating the performance of anomaly detection algorithms, while anomaly detection techniques can enhance VVUQ processes by identifying potential issues in models or data.

2.4.4 Modern VVUQ in the Context of SBDT

ML-based VVUQ approaches heavily rely on data. In the preceding sections, the thesis introduced several interfaces which may serve as suitable data sources. PM/OCED may provide event logs for VVUQ approaches —ML approaches can ingest those. PPC systems may provide additional data to enrich the event log. This section will shed light on a systematic literature review (SLR) on VV in the context of DTs. After summarizing the main findings, several VV approaches with increasing sophistication are presented. The section closes with a discussion of the most promising approaches.

(**Bitencourt2023**) conducted a SLR on VV in the context of DT —a relatively new field. They did not consider uncertainty quantification in their SLR. As (**hua2022validation**) note, there are no structured and established frameworks for validating DTs. This statement holds for all sectors where DTs are applied. The SLR analyzed 269 papers. They applied the 4R framework by (**Osho2022jmsy**) to describe the capabilities of the analyzed twin frameworks. The 4R framework consists of the four dimensions *Representation*, *Replication*, *Reality* and *No DT*. The SLR found that most frameworks (49%) rather developed a DM or DS. Another 26% of DT were only able to *represent* the physical entity. Highly sophisticated DT were the exception, not the rule. Considering this trend, VV may not be a topic researchers are interested in at first sight. (**Bitencourt2023**) identified a trend throughout the years of increasing modelling capabilities of the considered DT. Up to the year 2023, the trend is still increasing. They identify more data sources as the main driver for this trend. From the 269 papers, 47% have been applied in the manufacturing domain. Of all classified DT, one key insight is that most authors performed atleast one form of VV.

DTs ranking in the *reality* category where most often verified and validated. The authors deduct that because of the increasing complexity of the model, VV may become a stressing topic. (**Nie2023rcim**) propose a multi-agent and cloud-edge orchestration framework leveraging Digital Twin and IIoT to optimize distributed manufacturing resources. Their framework integrates a Convolutional Neural Network (CNN) with a bidirectional Long Short Term Memory (LSTM) module to perform scheduling and self-adaptive strategies for intelligent, real-time production control. They compare their network with an earlier adopted algorithm and its result to perform VV. Quantifying the validity, they use Mean Squared Error (MSE) and Mean Absolute Error (MAE), see **??**. This can still be considered manual VVUQ, because they did not use an automatic algorithm to compare the metrics of both models. PROOF (**Lv2023rcim**) defined target

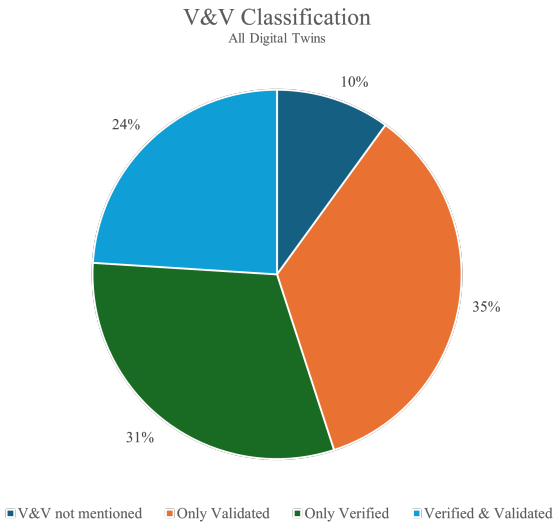

Figure 2.5: Donut chart showing the distribution of VV methods in the context of DT.

scenarios and boundaries to verify their DT for fault diagnosis of a drilling platform. They defined intervals for the metrics to be valid. Validation was performed by conducting a case study. Latter can be termed qualitative validation because the factual similarity of the case study against the DT has been assessed. Regarding their verification approach, they go one step further than (**Nie2023rcim**) by defining control intervals. This is somewhat automatic verification, but still requires human intervention when the values leave the defined intervals. (**Chen2023**; **Mykoniatis2021jim**) both performed visual inspection to verify their DT. Both used cameras or other visual inspection tools to compare the DT results with reality. (**Chen2023**) used thermal cameras to conduct temperature scans and compared the results with temperature prediction of the twin. (**Mykoniatis2021jim**) used video recordings to validate their twin behaviour, but consulted KPIs to further validate their twin. Both approaches use aids for measurements and visual inspection. The measurements still have to be conducted and compared manually. Some authors performed statistical VV, although the measurements were not automated. (**Wang2023asoc**) compared historical data with the twin data and calculated the mean absolute percentage error of both datasets, **??**. For validation they relied on conducting a case study, thus not fully automating VV. (**Min2019ijinfomgt**) performed verification using PPC KPIs, automating the VV process even further. Their validation included using a validation set and measuring a set of metrics including error of fit and accuracy. This approach can be considered the first step towards a fully automated VVUQ process. (**Bitencourt2023**) conclude that the majority of the analyzed papers performed VV manually, with only a few authors automating the process. They also note that most authors did not consider uncertainty quantifi-