Digital Twin-based Cyber-Attack Detection Framework for Cyber-Physical Manufacturing Systems

Efe C. Balta, Michael Pease, James Moyne, Kira Barton, Dawn M. Tilbury

***Abstract*—Smart manufacturing (SM) systems utilize run-time data to improve productivity via intelligent decision-making and analysis mechanisms on both machine and system levels. The increased adoption of cyber-physical systems in SM leads to the comprehensive framework of cyber-physical manufacturing systems (CPMS) where data-enabled decision-making mechanisms are coupled with cyber-physical resources on the plant floor. Due to their cyber-physical nature, CPMS are susceptible to cyber-attacks that may cause harm to the manufacturing system, products, or even the human workers involved in this context. Therefore, detecting cyber-attacks efficiently and timely is a crucial step towards implementing and securing high-performance CPMS in practice. This paper addresses two key challenges to CPMS cyber-attack detection. The first challenge is distinguishing expected anomalies in the system from cyber-attacks. The second challenge is the identification of cyber-attacks during transient response of CPMS due to closed-loop controllers. Digital twin (DT) technology emerges as a promising solution for providing additional insights into the physical process (twin) by leveraging run-time data, models, and analytics. In this work, we propose a DT framework for detecting cyber-attacks in CPMS during controlled transient behavior as well as expected anomalies of the physical process. We present a DT framework and provide details on structuring the architecture to support cyber-attack detection. Additionally, we present an experimental case study on off-the-shelf 3D printers to detect cyber-attacks utilizing the proposed DT framework to illustrate the effectiveness of our proposed approach.**

***Note to Practitioners*—This work is motivated by developing a general-purpose and extensible digital twin-enabled cyber-attack detection framework for manufacturing systems. Existing works in the field consider specialized attack scenarios and models that may not be extensible in practical manufacturing scenarios. We utilize digital twin (DT) technology as a key enabler to develop a systematic and extensible framework where we identify the abnormality of a resource and detect if the abnormality is due to an attack or an expected anomaly. We provide several remarks on how our proposed framework can extend existing industrial control systems (ICS) and can accommodate further extensions. The presented DTs utilize data-driven machine learning models, physics-based models, and subject matter expert knowledge to perform detection and differentiation tasks in the context of expected anomalies and model-based controllers that control the manufacturing process between multiple setpoints. We utilize a model predictive controller on an off-the-shelf 3D printer to run the process, and stage anomalies and cyber-attacks that are successfully detected by the proposed framework.**

This work was funded in part by NSF 1544678 and NIST Award No.70NANB19H090.

Efe C. Balta is with the Automatic Control Laboratory, ETH Zurich, 8092 Zurich, Switzerland. ebalta@control.ee.ethz.ch

Michael Pease is with the National Institute of Standards and Technology, Gaithersburg, MD 20899, USA michael.pease@nist.gov

James Moyne, Kira Barton, and Dawn M. Tilbury are with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109, USA {moyne,bartonkl,tilbury}@umich.edu

I. Introduction

Smart manufacturing (SM) is an increasingly important paradigm that promotes the use of run-time and historical data collected via onboard and additional Internet of Things (IoT) sensing in the manufacturing system to derive decisions for the plant floor [1]–[4]. Plant floor decisions include production scheduling and dispatch, predictive maintenance, anomaly detection, and process control. The decisions are implemented, often in run-time, on the resources in the manufacturing system to minimize disruptions, by integrating cyber and physical systems in modern manufacturing resources, allowing them to be reconfigurable and robust in response to disturbances. This framework of data-enabled decision-making coupled with cyber-physical manufacturing resources is commonly referred to in the industry as Cyber-Physical Manufacturing Systems (CPMS) [5], [6].

As CPMS become more complex, supporting decision-making processes becomes increasingly challenging. Additionally, decision-making logic designed for the nominal conditions of a CPMS may underperform or fail to detect certain abnormalities in the system due to complex interdependencies between multiple resources in a manufacturing process [7], [8]. Another important implication of the cyber-physical nature of CPMS is its vulnerability to cyber-attacks. As cyber components are linked to their physical counterparts, attacks that are initiated in the cyber domain may cause harm and damage to the physical manufacturing resource, product, or even the human workers that are interacting with the CPMS [9]. However, detecting cyber-attacks through traditional IT-based attack detection technology deployed on or in operational technology (OT) devices and environments can sometimes adversely impact OT performance or safety. Therefore, new and effective methods to monitor CPMS and detect cyber-attacks are required.

Detecting cyber-attacks on CPMS is not a trivial task for several reasons. Systems routinely undergo faults and expected abnormalities, namely, physical degradation, anomalies. These anomalies may be hard to distinguish from a carefully targeted cyber-attack (e.g., one with malicious intent) as these attacks often mimic the expected anomalous behavior to deceive the decision-making logic. Furthermore, cyber-attacks may originate from non-malicious intent (e.g., mis-calibration, version

mismatch, malfunction, etc.), which also causes difficulties in distinguishing them from anomalies. Additionally, run-time process controllers can be updated with new firmware and undo changes to setpoints and resource control inputs, making cyber-attack detection even more challenging.

To address the issue of monitoring CPMS, digital twin (DT) technology has emerged as a fundamental tool for *twinning* physical resources to provide additional analysis capabilities and delivering insights on the run-time system in addition to the as-designed conditions. The potential and flexibility of DT technology has generated significant research interest from academia and industry on applying DTs for supporting SM in practice, hailing DTs as the cornerstone technology for realizing SM [2], [10]–[13]. Building on the adoption of DT for supporting SM, the main contribution of this work is to propose a DT-based method to address the challenge of *cyber-attack detection for CPMS*.

Specifically, in the context of this work, a DT is a software replica of a physical counterpart (i.e., the physical twin), system, process, or product, and has a purpose of impacting an aspect of the physical twin and its environment in a positive way through utilizing models, data analytics, and subject matter expertise (SME) [2], [14]. A DT implementation consists of one or more compute resources as required to meet scalability, modularity, and maintainability requirements. Use of a single DT (i.e., one compute resource) for complex CPMS has been proposed [5], [12], [15]. However, scalability, modularity, and maintainability of such solutions often becomes a challenge in practice. More recently, a framework of multiple DTs that utilize structured abstractions to improve scalability, flexibility, maintainability, and modularity of DT-based solutions has been proposed [2], [14], [16], which is the DT architecture adopted in this work. The DT framework presented here utilizes multiple compute resources to distribute different data collection and analysis tasks supporting the anomaly and cyber-attack detection processes in a flexible, modular, and reconfigurable fashion. Therefore, we focus on a framework-based method that differs from the single DT approaches, discuss the benefits of our design choice throughout the paper, and highlight how this approach encapsulates many others in various ways. As DTs themselves are software entities, they may also bring along the additional burden of vulnerabilities that could compromise the physical components through cyber-attacks. In this work, we focus on utilizing DTs to support cyber-attack detection rather than considering the cyber-security of the proposed DTs themselves.

Traditional enterprise cybersecurity control implementations are not always possible or feasible within Industrial Control Systems (ICS) network environments and improper implementations can have unintended consequences [17]. Typically, passive monitoring capabilities for supporting threat detection within ICS network environments are implemented as risk management strategies within these networks. However, countering the growing threats facing ICS environments requires both passive and active monitoring [17]. These capabilities applied at the lowest levels could be utilized to detect signs of anomalous behavior resulting from cybersecurity threats.

The existing literature has focused on cyber-attack detection and mitigation for CPS by leveraging cross-domain knowledge [18]–[21]. Furthermore, side-channel information and data-driven models have been often utilized to detect certain types of cyber-attacks [22]–[26]. In many of these proposed methods, a training data set is utilized to generate an underlying process signature, which is then compared against the run-time measurements to detect cyber-attacks. While highly effective methodologies for cyber-attack detection are presented, they are often customized for a specific system or operation and thus do not provide ways to extend the proposed frameworks to be adapted for a variety of CPMS in a modular flexible way.

Most of the existing literature demonstrates a specific method and implementation scenario. For a holistic cyber-security approach, an extensible and modular framework where various components can be extended and modified is needed. See [27] for a survey of cybersecurity approaches in CPMS, where the importance and need of a framework for the security of the system components and the control architecture is highlighted along with examples from the literature. To this end, a digital twin-based framework, such as the one proposed in this work, enables a unifying approach where solutions from the literature may be efficiently implemented. Additionally, utilizing our framework, solutions from the literature may be extended with novel capabilities such as distinguishing cyber-attacks from expected anomalies and performing transient analysis for controlled processes between varying setpoints, which have not been considered in the existing literature.

Many of the aforementioned methods on CPS and CPMS cybersecurity from the literature are often referred to as physics-based attack detection methods (see [28] for a detailed survey). We note that many of the methods we employ for attack detection in our proposed work utilize physics-based attack detection methods in their core. Therefore, we consider physics-based attack detection methods as complementary to our work and note that the proposed framework in this paper enables the implementation of such methods with further extensions. Most notably, the majority of the existing literature considers the cyber-attack detection problem for a CPMS with no anomalies, which is often unrealistic in practical scenarios. Additionally, most existing methods in the literature rely on threshold-checking on the residual signals, which may underperform for controlled processes with setpoint changes during transients. We propose novel approaches to overcome these challenges in our proposed framework and methods.

Recent work provides a methodology to detect and differentiate specific types of cyber-attacks from equipment failure [29]. The method in [29] utilizes specific models and assumptions, which may be difficult to extend and scale for a general CPMS with various types of attacks. Therefore, there exists an opportunity to address the aforementioned shortcomings and support both manufacturing and cybersecurity automation enhancements by leveraging common technological enablers such as DT and the Industrial Internet of Things (IIoT). DT and IIoT could be tools that afford us with an opportunity to address cybersecurity issues from a generic and reusable perspective.

Previous research, such as [30], demonstrates techniques

to utilize Industry 4.0 technologies and methodologies such as IIoT, Industrial Internet of Services (IIoS), and DTs to create smart factories and establish "Knowledge as a Service" manufacturing processes to monitor product or service quality. Our research builds on the previous literature and investigates utilizing cybersecurity DT technology to monitor devices and processes for abnormal conditions that could be indicators of cybersecurity events in the context of run-time controller inputs and anomalies. These cybersecurity DTs could be implemented to support a passive/active hybrid approach to protect the ICS environment from advanced device-level risks.

Our method is capable of working with existing architectures for anomaly detection in industrial systems and enables scalability to multiple resources in a CPMS thanks to its DT-centric design. The contributions of this work are:

- An extensible DT-based solution framework for cyber-attack detection in cyber-physical manufacturing systems, capable of integrating with existing solutions in practice.
- A methodology to distinguish cyber-attacks from expected anomalies for a controlled cyber-physical system.
- A novel experimental demonstration of the proposed *DT-based* method on an off-the-shelf 3D printer to demonstrate the effectiveness, flexibility, and scalability of the proposed approach.

The rest of the paper is organized as follows. Section II provides preliminary definitions of concepts used in this work and provides the formal problem definition. Section III provides the framework architectures for the CPMS with all of the proposed DTs for cyber-attack detection. Section IV presents the proposed DT-based cybersecurity approach and Section V demonstrates the experimental implementation and results. Additionally, we provide remarks on how solutions from the literature can be implemented by utilizing our proposed framework, which provides a guideline for design and implementation. Concluding remarks and future research directions are presented in Section VI.

II. Preliminaries and Problem Statement

In this section, we first present definitions and background knowledge that will be useful in further discussions. Then, we formally state our problem in the context of the introduced formal concepts.

A. Classification of Abnormality Types

To address the challenge of cyber-attack detection for a CPMS, we first present a classification of anomalies, attacks, and faults in the context of the present work. Figure 1 presents various types of attacks and anomalies for a CPMS resource. Each item in Fig. 1 that is inside the box (output measurable effect on the system) is an event that results in an effect on the physical process that is categorized as the corresponding set (e.g., a failed sensor event results in a fault that is a subset of anomalies and output measurable abnormalities). The representation in Fig. 1 is inspired by [7], where types of anomalies and faults for smart manufacturing systems and their detection mechanisms are discussed in detail.

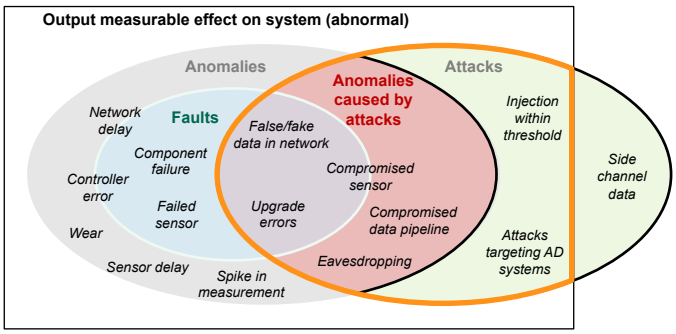

Fig. 1. Illustration of the subspaces for observable abnormalities, anomalies, faults, and attacks considered in this work. The scope of this paper is outlined with orange borders in the figure. AD: Anomaly detection.

Definition 1 (Anomaly [7]). *An occurrence that is different from what is standard, normal, or expected.*

Definition 2 (Fault [7]). *An anomaly that is related to an unwanted situation and may be associated with failure, malfunction, or quality degradation.*

Thus, an *anomaly (fault) detection* (A(F)D) mechanism detects the result or onset of an anomaly (failure) event. Some events such as network delays and controller errors may result either in failures that would be classified as faults, or only anomalous behavior that does not necessarily result in failure.

Definition 3 (Cyber-attack [31]). *The realization of some specific threat that impacts the confidentiality, integrity, accountability, or availability of a computational resource.*

The "normal" behavior of the system is defined by the nominal operation of the system without any anomalies or cyber-attacks and we use the term *abnormal* for all other system behavior. Thus, we say that a system is abnormal if its output measurements are not consistent (evaluated by a classifier) with the measurements from the nominal operation without any anomalies or cyber-attacks. Additionally, we say an input has an *output measurable effect on the system* if given sufficient measurements of the output, it is possible to determine the effect of a (possibly exogenous) input on the output, which could be immediate or at a later time. For the rest of the paper, we simply use the term *attack* when referring to a cyber-attack to lighten the terminology, unless specified otherwise. The term cyber-attack is also used in certain discussions to emphasize the context.

B. Problem Statement

The set of attacks depicted in Fig. 1 has three distinct sub-spaces; attacks that are not output measurable (e.g., side-channel attacks), attacks that are output measurable but do not necessarily cause anomalies, and attacks that are output measurable and cause anomalies. Within the scope of this work, we are focusing on attacks that have output measurable effects on the system. Thus, the goal of our proposed DT is to detect the aforementioned output measurable attacks. Further discussions on the attack intent and attack types that are not output measurable are beyond the scope of this work. We

note that, for example, the large literature of physics-based attack detection methods [28] and related works fall under this category. Since anomalies are not necessarily caused by attacks, an effective methodology should be able to distinguish anomalies caused by attacks from the inherent anomalies that we expect to see in the system, which we term as *expected anomalies* in further discussions.

Remark 1. *Within the context of cyber-attacks on CPMS, we do not necessarily require malicious intent. For example, we consider a miscalibrated sensor as a non-malicious attack.*

Additionally, we note that the physical system is a controlled CPMS resource, thus the operational characteristics of the system may be modified by a controller. This results in transient behavior and multiple setpoint references that must be analyzed in run-time to mitigate false-positives in attack detection. Furthermore, due to our assumption on the presence of anomalies in the system, expected anomalies and attacks may be *inseparable* in the output as they may result in similar output effects.

Remark 2. *We note that our work differs from the past literature as we do not rely on a specific system model or analysis tool to provide our results. Instead, we present a general-purpose DT framework where data-driven and physics-based information about the CPMS may be utilized efficiently to detect cyber-attacks in an extensible and systematic manner.*

We formally state our problem as *"How do we develop an effective methodology to identify attacks with output measurable effects in the presence of anomalies in a controlled CPMS?"* In this work, we propose a DT-based method to address this problem and present case studies to illustrate our proposed method.

III. Proposed DT-Based Methodology

In this section, we present the proposed methodology to utilize DTs for attack detection in the context of anomalies and controllers in the system. We start by introducing the framework architecture with some of the existing DTs that may already be in place. We discuss how related methodologies from the literature can be implemented by the proposed DTs in our architecture.

A. Framework Architecture

Figure 2 illustrates the architecture of the controlled CPS framework with the proposed DTs considered in this work. To avoid confusion of terminology, we use the term *process* instead of *system* in this section (e.g., a physical system is a physical process). The physical process on the top block is the manufacturing process we control and analyze utilizing the proposed framework. The physical process may be discrete or continuous depending on the application domain. The execution of discrete manufacturing processes is often considered in terms of *runs* where a single unit (or batch) is manufactured. We consider the data collected during the run as in-situ and the data collected after a run is completed as ex-situ (e.g., for post-process quality control). The framework

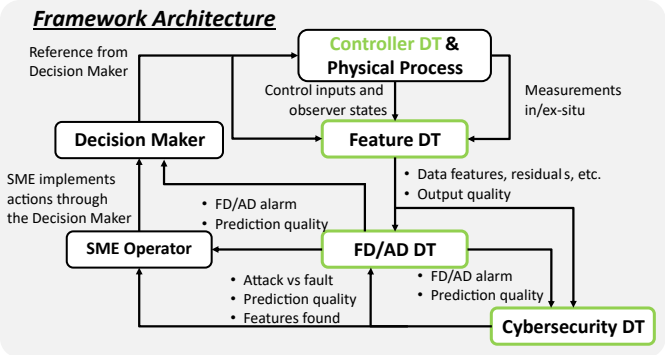

Fig. 2. The framework architecture including all the DTs and physical components. The architecture provides a basis for further extensions based on the needs on a certain physical process. The decision-maker in the architecture may be autonomous or purely advisory depending on the application domain. The color green indicates the DTs in the framework.

architecture presented in Fig. 2 is largely based on augmenting existing feature-based anomaly and fault detection systems in the literature (e.g., [16], [19], [32]–[34]).

Remark 3. *It is important to note that the abstraction of what constitutes a DT is a design decision. We adopt the DT framework methodology instead of considering a single DT entity, e.g. [15]. However, there is no loss of generality since the proposed DTs in this work may be encapsulated by a single DT. As previously discussed, we utilize the framework approach, following [14], [34], [35] to improve flexibility, interoperability, and maintainability of the proposed method. Therefore our approach is complementary to recent standards, such as [15], and proposes a new DT-based approach for the cyber-attack detection problem, without loss of generality in the context of the existing works.*

In the rest of this section we provide details of the blocks in Fig. 2 and present definitions, purpose, assumptions, inputs, outputs, and possible extensions for the proposed DTs.

1) Physical Process: We assume that the physical process (referred to as *process* for the rest of the paper) is a manufacturing process that has sensors in place to collect in- and ex-situ data and the measurements are available to the DTs in the framework for data analysis purposes in run-time as well as in the form of historical data through a database. A discrete-time representation of the process is then given in a general form as

$$\boldsymbol{x}(t+1) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)) \tag{1a}$$

$$\boldsymbol{y}(t) = g(\boldsymbol{x}(t), \boldsymbol{v}(t)), \tag{1b}$$

where $\boldsymbol{x}(t)$ denotes the process states, $\boldsymbol{u}(t)$ is the process input, $\boldsymbol{y}(t)$ is the measurement, $\boldsymbol{w}(t)$ and $\boldsymbol{v}(t)$ are process and measurement noise respectively, t is the discrete-time index, and f and g are the process and measurement models, respectively. The time index t represents either the in-situ time index or the run-to-run (R2R) index based on the applications of interest. We can augment the notation of (1) to include both time indices if needed (e.g., $\boldsymbol{x}_j(t)$ where j is the R2R index and t is the in-situ time index). Note that $\boldsymbol{y}(t)$ may represent an in- or ex-situ measurement in this context and it is referred

to as the process *output* when the interpretation is clear from the context.

An SME monitors the process through the DTs in the framework as illustrated in Fig. 2, and implements reconfigurations or changes to the process through a decision-maker. We assume that a decision-maker exists in the framework without loss of generality. The decision-maker provides setpoint references $\boldsymbol{r}(t)$ for the process to track (in the sense that $||\boldsymbol{r}(t) - \boldsymbol{y}(t)||$ is as small as possible in a suitable norm).

While we assume that the process has the form in (1) for our further discussions and developments, systems of various forms and dynamics can be considered here (e.g., discrete event systems). Additionally, the process itself can be modeled as a separate DT to perform simulation-based analysis on the process.

2) Controller DT: The Controller DT houses the run-time controller with the control logic, as well as observers, process models, and simulation tools. The Controller DT employs various control methods and logic (e.g., feedback, feedforward, rule-based, hybrid, etc.) to regulate the process measurements $\boldsymbol{y}(t)$ toward the reference setpoints $\boldsymbol{r}(t)$ provided by the decision-maker. To perform state-based control, the Controller DT may incorporate various types of filters and estimators to estimate the current and future states of the process by using the measurements and information such as historical data, or model adaptation information provided by other DTs in the framework (e.g., models of the noises $\boldsymbol{v}(t)$ and $\boldsymbol{w}(t)$). Control inputs $\boldsymbol{u}(t) \in \mathcal{U}$ are implemented on the process, where $\mathcal{U}$ denotes an input constraint set. In practical implementations, there may be additional safety control loops that bypass the control input implementation (e.g., emergency stop switch for a robotic manipulator).

- The inputs are the reference setpoints $\boldsymbol{r}(t)$ from the decision-maker, and run-time data (sensor measurements including but not limited to $\boldsymbol{y}(t)$) from the process.
- The outputs are the control input $\boldsymbol{u}(t)$ to be implemented on the process, states of the process (estimated via observers), process indicators, and model states considered by the control logic, and measurement signal $\boldsymbol{y}(t)$.

The Controller DT may utilize information from other DTs, e.g., to simulate system dynamics for what-if analyses of the physical process [34] or estimate remaining useful life to detect anomalies and optimize end-of-life control strategies [16].

3) Feature DT: The Feature DT provides uniform data streams to the DTs in the framework to improve the interoperability of the framework. Existing run-time anomaly detection methods often rely on residual analysis to provide threshold-based decisions [7], [32]. We assume that an SME defines the desired residual signals with specific features, and implements them as part of the Feature DT so that the residual information is shared with other DTs for further data analysis.

Another important task of the Feature DT is to evaluate key process indicators (KPIs) for the process. Various types of KPIs include health indicators, performance indicators, and efficiency indicators [34], [36]. Similarly, the Feature DT may be tasked to pre-process or partition large scale or high sampling-rate measurement data for another DT that performs statistical learning on the measurement data.

- The inputs are the data streams from the decision-maker, process, and the Controller DT. There inputs are aggregated, and pre-processed by the Feature DT.
- The outputs are the processed data streams with the SME designed data features, residuals, and an indication of output quality whenever applicable.

In many practical applications, the physical process and its Controller DT are on a different interface and platform than the data analytics platform. In such cases, the Feature DT is tasked with implementing the appropriate interfaces for data communication and storage to a local database. While the Feature DT is implemented in the framework based on the specific needs of other components and DTs, we also leverage existing Feature DTs, if available.

4) FD/AD DT: The FD/AD DT performs fault and anomaly detection on run-time data streams. Preliminary detection capabilities are included in most CPS for reliable run-time performance. Such detection mechanisms are considered as part of the FD/AD DT here. The FD/AD DT is usually built to perform threshold-based limit-checking on the physical process. The FD/AD DT may include safety monitoring and performance monitoring systems to detect anomalies and faults. A review of various model-based anomaly detection methods for control systems is given in [32] and more specific anomaly types for smart manufacturing systems with possible detection methods are discussed in [7].

- The FD/AD DT takes inputs from the Feature DT to perform its analysis. Historical data provided by a database may also be utilized for analysis. Additionally, the attack detection predictions of the Cybersecurity DT may be utilized to refine threshold parameters in the FD/AD DT.
- The FD/AD DT provides its outputs for the detection of an anomaly with an indication of prediction quality (i.e., confidence in detection) to the decision-maker and the operator in the system. We further utilize the outputs of the FD/AD DT to implement our Cybersecurity DT. The FD/AD DT may also share the corresponding data traces for the predicted faults and anomalies.

While we assume a threshold-based limit checking method for the FD/AD DT here, additional methods that adapt and learn anomalous or fault behavior of the process over time may be implemented as extensions.

5) The Cybersecurity DT: The Cybersecurity DT provides predictions about attacks on the system in the context of anomalies and transient response of the controlled process. We assume that the Cybersecurity DT is designed by an SME knowledgeable on the cybersecurity of the process and we focus on attacks with output measurable effects as stated earlier. In the absence of such prior knowledge, historical data may be used to understand the normal system behavior initially. In this context, abnormalities can be recorded during operation and labeled as normal, anomalous, or attack data by an SME. If an SME or enough historical data is not available to initialize the framework, the proposed approaches may not be applicable. The Cybersecurity DT is a novel contribution of this work to distinguish cyber-attacks from expected anomalies for a controlled process, and we provide a detailed analysis of the Cybersecurity DT in later sections.

- The output data streams and FD/AD indications of the Feature DT and the FD/AD DT are the inputs for the Cybersecurity DT. Additional historical data available through a database is also used as inputs for training data models.
- The predictions of attacks versus expected anomalies with an indication of the prediction quality and key features found in the analyzed signal (e.g., features indicating the type and/or source of an attack) are the outputs of the Cybersecurity DT. Additionally, the attack features found in the analyzed data are shared with the FD/AD DT and the SME Operator for further analysis.

6) SME Operator: The operator monitors the outputs of the FD/AD DT and the Cybersecurity DT to further analyze if the physical process has an anomaly or is under a cyber-attack. For this purpose, the DTs report their prediction quality and the features found in the data so that a human SME may further investigate any abnormalities.

7) Decision Maker: The role of the decision-maker is to provide an interface between the SME and the plant floor. Many CPMS in practice utilize a supervisory control and data acquisition (SCADA) layer as a decision-maker. The decision-maker may have a supervisory role where it takes actions on the plant floor by making autonomous decisions. If the decision-maker is purely advisory, the SME may implement actions and prescribe references directly to the controlled plant, bypassing the decision-maker. In our context, the decision-maker provides details and updates on the reference signal $\boldsymbol{r}(t)$ for the process.

The presented framework forms a basis for the analysis of cyber-attacks for CPMS in the context of closed-loop controllers and expected anomalies. The proposed DTs in our framework are building blocks that can be implemented via various detection and classification methods from the literature. The DTs therefore provide valuable abstractions for performing attack detection in CPMS via a composition of methods, illustrating how the proposed framework is general enough to accommodate various methods and solutions from the literature (e.g., [8], [19], [20], [29], [33], [37], [38]). See Section V-F for further discussions. Furthermore, the presented framework may be aggregated into a system-level DT that operates within or outside of the four walls of operation (e.g., at the supply chain level) [2], [39].

IV. The Cybersecurity DT

The architecture of the Cybersecurity DT is illustrated in Fig. 3. The Cybersecurity DT utilizes a Detector DT and a Consistency DT to analyze run-time and historical data, and perform online data analysis. In this section, we present the architecture of the Cybersecurity DT as one of the main contributions of our work. Then, to illustrate the utility of the proposed Cybersecurity DT, we provide details of a proposed attack detection method to distinguish attacks in the context of anomalies for a controlled process. Throughout the section, we highlight how various other methodologies from the literature may be utilized in our proposed framework, wherever appropriate.

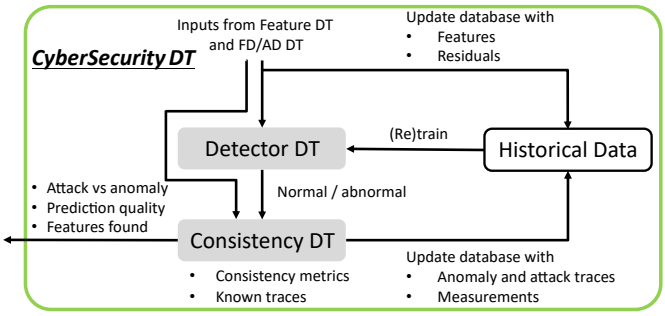

Fig. 3. The architecture of the Cybersecurity DT. The Detector DT and the Consistency DT are used for detecting abnormalities and attacks on the physical process. The historical data is stored in a database for model training as well as knowledge storage and SME data mining of the types of expected anomalies, attacks, etc.

A. Cybersecurity DT Architecture

We propose a Cybersecurity DT that utilizes two DTs to perform abnormality detection and attack detection, so that it can predict the presence of a cyberattack on the physical process. The Cybersecurity DT also has a database that includes historical data that is used for model training, data mining, and data analysis. Note that while the Cybersecurity DT uses run-time data to predict attacks, the DT may or may not run synchronously with the physical twin. We assume that the data streams within the framework are time-stamped such that asynchronous DT predictions that indicate predicted time-instance of an attack onset are possible. The actual time frame of the DT versus the physical process for a practical implementation depends on the application domain of the process.

1) Detector DT: Noting that our goal is to detect attacks that have output measurable effects, we first need to identify if a measurement is abnormal. The Detector DT is tasked with performing abnormality detection on the process data by leveraging the anomaly prediction from the FD/AD DT. A key problem with anomaly and attack detection is the scarcity of abnormal process data versus the abundance of *normal* process data, leading to an unbalanced data set. For this purpose, machine learning models such as one-class discriminators [38], [40]–[42] and auto-encoders [43]–[45] are often utilized in the literature for abnormality detection to represent the normal data sufficiently well in a projected space such that abnormal data can be detected efficiently (see Section IV-B1).

Data-driven models may be utilized in the Detector DT and with the assumption on the availability of sufficient normal process data to train data-driven models. Additionally, the FD/AD DT predictions may be utilized to improve the abnormality detection in the Detector DT. We present a Detector DT leveraging one-class discriminators in later sections and demonstrate our approach in the experimental study.

- Inputs to the Detector DT are the outputs from the Feature DT (e.g., features of the process measurements) and the FD/AD DT (e.g., prediction of an anomaly, anomaly traces found in the analyzed data). Historical data from a database is used to train data-driven models in the Detector DT.

- The Detector DT provides an indication of abnormality for the processed data. If the abnormality detection has an associated detection threshold, the threshold value is reported as well.

If the physical twin undergoes modifications that affect the dynamics of the process (e.g., physical wear, maintenance event, re-calibration of sensors, new data streams, software updates), the Detector DT may be re-trained given that sufficient data in the historical database is available, adjusted to another model in its library (if the new context environment has already been modeled), or modeled to track certain dynamics such as slow drift as the new normal behavior. We focus on data-driven discriminator based abnormality analysis in our work. Extensions to the presented approach include physics-based analysis utilizing models such as (1), residual-based analysis (e.g., using a *golden trace*), and rule-based analysis e.g., various statistical process control methods [46].

2) Consistency DT: If a measurement is labeled as abnormal by the Detector DT, or alternatively if an anomaly is detected by the AD-DT, further analysis is performed by the Consistency DT. To understand if an abnormal measurement is due to an attack or an anomaly, we utilize the notion of *consistency metrics* on the process data. A consistency metric for the physical process (1) characterizes the expected behavior of the system during expected anomalies, e.g., how a measurement signal changes due to expected mechanical wear. We provide a formal definition of consistency metrics and how they are used for attack detection in Section IV-B2.

We assume that specifications defining the behavior of the system under *expected anomalies* are provided to the Consistency DT. Therefore, the Consistency DT monitors these specifications on the run-time process data to detect inconsistencies that predict the presence of an attack on the process. We focus on formal methods-based approaches to encode specifications for expected anomalies for the CPMS process. Within this context, the specifications are represented in terms of the progression of consistency metrics in a formal language that is used by the Consistency DT to monitor the process for attacks.

- Outputs from the Feature DT, Detector DT, and the FD/AD DT are utilized for evaluating consistency metrics and monitoring the metrics in run-time. Process specifications based on the consistency-metrics for expected anomalies are also provided to the Consistency DT.
- The Consistency DT outputs the prediction that an attack has occurred with the prediction quality and the features found in the data traces. The anomaly and attack traces found by the Consistency DT are also stored as part of the historical data for further analysis.

We assume an expert analyzes historical data of the process to evaluate consistency metrics and specifications for expected anomalies. Therefore, a new anomaly (one that is not considered in the set of expected anomalies) is predicted to be an attack by the Consistency DT. As the Consistency DT reports the features found on the data as well as data traces, an SME may design additional consistency metrics and specifications for a new anomaly by analyzing the process data. We present example consistency metrics and corresponding formal specifications for a CPMS process in the experimental study. Extensions of our approach could utilize data-driven methods to identify consistency metrics and corresponding formal specifications. Developing such methods is a subject for future work.

3) Historical Data: The historical database stores process data, the expected anomaly features, and data traces, as well as historical outputs from the Feature DT and the FD/AD DT. The database is updated with the outputs of the DTs for further analysis. Additionally, the SME updates the labels of the historical data to account for new expected anomalies encountered on the process. This procedure may be initialized with sufficient historical data to build consistency metrics. The SME can monitor abnormal data identified by the DTs to build a library of expected anomalies and better consistency metrics over time.

B. Proposed Illustrative Methods for Attack Detection

We present the theoretical background for the proposed illustrative detection methods used in the Detector DT and the Consistency DT for abnormality detection and consistency metrics based attack detection, respectively, in this section.

1) Abnormality detection: To implement abnormality detection, the Detector DT is trained on the historical process data $D = \{\boldsymbol{y}(t), \boldsymbol{x}(t), \boldsymbol{u}(t), \eta(t) \mid t = t_0, t_0+1, \ldots, t_0+n_w\}$, where $\eta(t) \in \{0, 1\}$ is a label for abnormality of a data point, to recognize features of normal process data. Note that in practice for an unbalanced dataset, we utilize only a single class label and define everything else as abnormal (i.e., η becomes trivial as all data in D corresponds to a single class). We denote the *normal* data boundaries trained by the Detector DT using the data D as $\mathcal{B}(D) \subset \mathcal{F}$, where $\mathcal{F}$ is a possibly nonlinear feature space where the Detector DT operates.

The Detector DT utilizes its trained model $\mathcal{B}(D)$ to monitor run-time data provided by the Feature DT and FD/AD DT and detect if current measurements of the physical process are normal, i.e., if $\psi(\boldsymbol{y}(t')) \in \mathcal{B}(D)$, where $\psi : \mathcal{Y} \to \mathcal{F}$ is a map from the measurement space $\mathcal{Y}$ to the feature space $\mathcal{F}$ of the Detector DT. Based on this analysis the Detector DT outputs its prediction as a label $\hat{\eta}(t') \in \{0, 1\}$ of normal versus abnormal. Additionally, probabilistic predictions and prediction quality measures may be provided. To this end, statistical learning methods such as the ones provided in [26] may be utilized. Alternatively, physics-based methods [28] may be utilized to detect abnormalities.

The training for the Detector DT utilizes historical process data of the steady-state operation at a predefined setpoint reference, e.g., $\boldsymbol{r}(t) = \bar{\boldsymbol{r}}$, to train $\mathcal{B}(D)$. However, the setpoint of the process may be altered either by a decision-maker or by a closed-loop controller on the physical process. The setpoint changes result in transient dynamic behavior on the system (1), which may cause false positives by the Detector DT.

To mitigate false positives of the Detector DT during transients, we utilize the solution map of the process (1), $\phi : \mathcal{X} \times \mathcal{U}^\infty \times \mathbb{Z}_+ \to \mathcal{X}$, where $\mathcal{X}$ is the state space of (1) and $\mathcal{U}^\infty$ is the space of sequential control inputs on (1).

Given an initial state $\boldsymbol{x}(t_0)$ and a control sequence $\boldsymbol{u} \in \mathcal{U}^{\infty}$ over a time interval including the interval $[t_0, t_c]$, we have

$$\phi(\boldsymbol{x}(t_0), \boldsymbol{u}; t_c) = \boldsymbol{x}(t_c), \tag{2}$$

where $\boldsymbol{x}(t_c)$ is the state at time t_c (i.e., the current state). Our motivation for the proposed abnormality detection method is to utilize the trained data boundaries $\mathcal{B}(D)$ during transient response. Roughly speaking, as $\mathcal{B}(D)$ is trained for the process at a given setpoint, we define a projection using ϕ to estimate state of the process at a previous setpoint given the transient observations (i.e., as the process moves away from the said setpoint) and the control inputs. If the process is normal, (i.e., no attacks or anomalies), the projected state should be within $\mathcal{B}(D)$.

Remark 4. *Forward projections of the set $\mathcal{B}(D)$ for the transient control inputs can also be used for abnormality detection. However defining such projections may in general be computationally expensive as $\mathcal{B}(D)$ may be control and state dependent, and new computations are needed at each control step. Therefore, we focus on the proposed projection type method for abnormality detection in this work.*

Formally, the goal of the Detector DT during transients is to estimate the initial state $\bar{\boldsymbol{x}}(t_0)$ of the process at time t_0 based on the observed sequence of states and control input $\boldsymbol{u}$ until the current time t_c. Let us denote the model of the state progression as

$$\Phi(\boldsymbol{x}(t_0)) = \begin{bmatrix} \phi(\boldsymbol{x}(t_0), \boldsymbol{u}; t_0) \\ \phi(\boldsymbol{x}(t_0), \boldsymbol{u}; t_0 + 1) \\ \vdots \\ \phi(\boldsymbol{x}(t_0), \boldsymbol{u}; t_c) \end{bmatrix}. \tag{3}$$

Additionally, let $\boldsymbol{x}$ denote the sequence of estimated states of the process between the times $[t_0, t_c]$. Then, the Detector DT solves the following minimization to estimate the initial state $\bar{\boldsymbol{x}}(t_0)$ by using the control input $\boldsymbol{u}$ and the state sequence $\boldsymbol{x}$.

$$\bar{\boldsymbol{x}}(t_0) = \min_{z} \{||\Phi(z) - \boldsymbol{x}||\}, \tag{4}$$

where z is an intermediate variable for the notation. For a normal process (i.e., process outputs with $\psi(\boldsymbol{y}(t')) \in \mathcal{B}(D)$), the solution of (4) is close (in the normed distance sense) to the actual initial state $\boldsymbol{x}(t_0)$. Therefore, the Detector DT evaluates the abnormality of the projected state $\bar{\boldsymbol{x}}(t_0)$ to evaluate the label $\hat{\eta}(t_c)$ for the current state $\boldsymbol{x}(t_c)$. Namely, if $\bar{\boldsymbol{x}}(t_0) \in \mathcal{B}(D)$, then the current state $\boldsymbol{x}(t_c)$ is predicted as *normal* by the Detector DT.

2) Consistency Metrics for Attack Identification: As mentioned earlier, the Consistency DT monitors the progression of a set of consistency metrics to understand if the abnormal data traces belong to a known anomaly. Since there may be many types of anomalies in the system, the design of the appropriate consistency metrics is often a challenging task. Following (1) we define the anomalous states and measurements as $\tilde{\boldsymbol{x}}(t)$ and $\tilde{\boldsymbol{y}}(t)$, respectively. Due to the nature of anomalies, true models of the anomalous process and measurements are often unknown, but we do have historical data of $\tilde{\boldsymbol{x}}(t)$ and $\tilde{\boldsymbol{y}}(t)$ for known anomalies. Let us define a combined run-time state as

$$\boldsymbol{\zeta}(t) = vec(\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{u}(t), \boldsymbol{r}(t)). \tag{5}$$

Our goal is to develop a consistency metric of type

$$\xi_i(t) = c_i(\boldsymbol{\zeta}; \boldsymbol{\theta}), \tag{6}$$

where $\boldsymbol{\theta} = \{(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{y}}_i, \boldsymbol{r}_i, \boldsymbol{u}_i) \mid i = 1, \ldots, n_h\}$, is the data set of length n_h from previous known anomalous process measurements, and $\boldsymbol{\zeta} = \{\boldsymbol{\zeta}(t), \boldsymbol{\zeta}(t-1), \ldots, \boldsymbol{\zeta}(t-n_w)\}$, for a window of size n_w. The Consistency DT monitors the progression of $\xi_i(t)$ with run-time data $\boldsymbol{\zeta}(t)$. Suppose we design our consistency metric such that $||\xi(t)|| \leq \delta(t)$, for some $\delta(t) \in \mathbb{R}$ during expected anomalies. Then, any abnormal measurement that results in $||\xi(t)|| > \delta(t)$ is logged as inconsistent. In our setting, an inconsistent measurement is a possible attack on the system. The design engineer for the consistency metric may utilize models of any kind (e.g., data-driven, physics-based, statistical, rules-based, etc.) to define a function c_i to evaluate consistency metrics. There may be multiple c_i in the system and the consistency DT may utilize an ensemble approach to detect inconsistencies.

A key challenge is defining $\delta(t)$ dynamically for a temporally measured signal. While specific $\delta(t)$ may be developed for individual use cases, the scalability of the design process becomes a prohibiting factor for using the proposed DT-based approach. An effective way to monitor consistency metrics may utilize signal temporal logic (STL) to develop logical predicates that prescribe the expected behavior of the measured signal over predefined measurement-time windows. STL is a widely used formalism to specify properties of a signal that is measured from a process. STL predicates for anomaly detection on an additive manufacturing (AM) process in a similar DT setting are presented in previous work [16]. We omit a detailed background on STL and refer interested readers to [47]. An STL formula π is formed by the following syntax:

$$\pi \triangleq \quad \top \quad | \quad p \quad | \quad \neg\pi \quad | \quad \pi_i \wedge \pi_j \quad | \quad \pi_i \, \boldsymbol{\mathcal{U}}_{[a,b]} \pi_j \tag{7}$$

where, $\top$ is logical true, p is a predicate, $\neg\pi$ is the logical negation of the proposition π, $\pi_i \wedge \pi_j$ is the logical conjunction of two propositions, and $\pi_i \, \boldsymbol{\mathcal{U}}_{[a,b]} \pi_j$ is the *until* operator defined as the proposition π_i being true at least until the proposition π_j is true in the time interval $[t+a, t+b]$, where t is the current time. A signal $\boldsymbol{s}(t)$ at time t is satisfied by a predicate p if $\ell(\boldsymbol{s}(t)) > 0$ for some function ℓ (i.e., $\boldsymbol{s}(t) \models p \iff \ell(\boldsymbol{s}(t)) > 0$). Here the operator $\models$ is used to indicate that the condition on the left side satisfies the condition on the right side. Additionally, $\bot = \neg\top$ is the logical false, the *eventually* operator is $\Diamond_{[a,b]}\pi \triangleq \top \, \boldsymbol{\mathcal{U}}_{[a,b]} \pi$, and the *always* operator is $\Box_{[a,b]}\pi \triangleq \neg(\Diamond_{[a,b]}\neg\pi)$.

We utilize signal temporal logic (STL) to encode the consistent temporal response of the system for expected anomalies. Let $\Pi = \{\pi_1, \ldots, \pi_{n_s}\}$ denote the set of consistency specifications to monitor. We want the process to satisfy all the specifications $\pi_i \in \Pi$, thus the Consistency DT monitors if the

conjunction of all specifications is satisfied (i.e., evaluates to true (top)). Thus, the consistency DT monitors the proposition

$$\boldsymbol{\zeta}(t) \models \bigwedge_{\forall \pi_j \in \Pi} \pi_j, \tag{8}$$

where we require the consistent run-time state measurement $\boldsymbol{\zeta}(t)$ (or a subset of the signals) to satisfy the conjunction of n_s propositions. While the proposed framework utilizes STL for consistency monitoring, extensions of the proposed framework may utilize various techniques including static and adaptive limit checking. We provide examples of inconsistency metrics and how they can be used for attack detection in the experimental demonstration, Section V.

To accommodate existing methods from the literature, attack detection algorithms relying on online monitoring of certain signal features and signatures [19], [22], [29], watermarks [21], [27], and statistical and model-based signal metrics [16], [20], for algorithmic decision-making, can be efficiently implemented via the proposed STL-based methodology in conjunction with the abnormality detection. In this case, the proposed detection methodologies may be utilized as part of the Detector DT, while consistency metrics may be implemented with the Consistency DT. Since STL is a convenient framework to monitor the behavior of threshold-based limit checkers, numerous other methodologies from the literature can be incorporated as part of the proposed framework. Additionally, we outline a generic consistency metric function and an STL based approach to detect inconsistencies, which can easily be used to implement established approaches from statistical process control (SPC) literature [46]. A discussion on the adaptation of a number of methods from the literature is given in Section V-F.

Statistical process monitoring methods such as average run length (ARL) may be utilized (see a similar use case in [29]) to tune the parameters and sensitivities as we further discuss in the case study section. Furthermore, the methodology recently presented in [29] can be implemented within our framework by utilizing their model-based detection methods as part of the Detector DT and using the Consistency DT to recognize the outputs of the multiple regressors in their framework.

We also emphasize that the aforementioned methods can be implemented as part of the Consistency DT itself as long as the inputs and outputs specified by the framework design are provided. The modularity of the proposed framework allows one to interchange and mix various solutions from the literature to develop a comprehensive solution framework for a holistic cybersecurity approach to CPMS.

As mentioned previously, unexpected anomalies, i.e., anomalies that are not considered in the set of anomalies during the design of the Consistency DT, are also predicted as inconsistent. When such new anomalies are encountered, the design engineer utilizes the new data traces $\boldsymbol{\theta}'$ containing data from the new anomalies to potentially design new consistency metrics and propositions to be monitored by the Consistency DT.

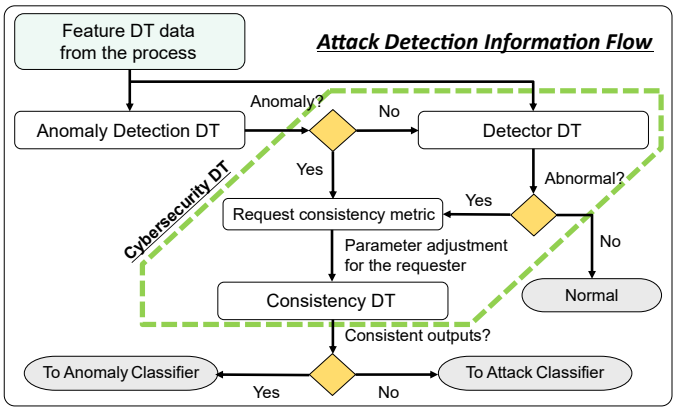

Fig. 4. Illustration of the information flow in the framework for attack detection with the Cybersecurity DT. The boundaries of the Cybersecurity DT are outlined with dashed lines.

C. Integration of the DTs for Attack Detection

Figure 4 illustrates the information flow between the DTs for attack detection in the presence of expected anomalies in the system. For the purposes of illustration, we denote a flowchart of how the prediction of the DTs inform each other and note that Fig. 4 does not illustrate the data shared between the DTs. The Feature DT continuously provides data to all other DTs in the framework. If the Feature DT includes an event trigger in its outputs, other DTs may use the trigger to perform analysis, or continuously perform analysis on the streaming process data.

First, the AD-DT performs threshold-based limit-checking to predict if there are any anomalies in the process. As we treat the AD-DT as part of existing detection mechanisms on the CPMS, we expect that its threshold limits are tuned at a desired operating characteristic by an expert. For our case study, we assume that the AD-DT has "wide" threshold limits set by an expert to reduce the false-positive rate. Consequently, if the AD-DT detects an anomaly, we conclude that the data is abnormal and move to request necessary consistency metrics from the Consistency DT.

If the AD-DT does not detect an anomaly, the Detector DT is utilized to predict abnormality. If there is no abnormality, the data is labeled as normal and no further action is taken. If the Detector DT predicts an abnormality, a consistency metric is requested from the Consistency DT. The consistency DT may utilize multiple consistency metrics with various parameter settings based on which DT requests a given metric. A consistent output is predicted to be an anomaly and this prediction is shared with an anomaly classifier or decision-maker for further analysis. If the data is inconsistent, an attack is predicted and the prediction is shared with an attack classifier or decision-maker for further analysis.

V. Experimental Demonstration on an Off-the-Shelf 3D Printer

In this section, we provide an experimental demonstration of our proposed DT solution on an off-the-shelf 3D printer as an illustrative CPMS resource. Experimental data collected from a printer under normal operation, anomalous operation,

and attacks are collected and analyzed by the Cybersecurity DT to present results of attack detection. The overview of the experimental setup is shown in Fig. 5. This experimental study is a demonstration of the proposed framework on a real-world setup. As previously discussed, our framework can be used in various application scenarios not limited to the one presented here and can be complementary to existing approaches. We discuss several examples from the literature at the end in Section V-F.

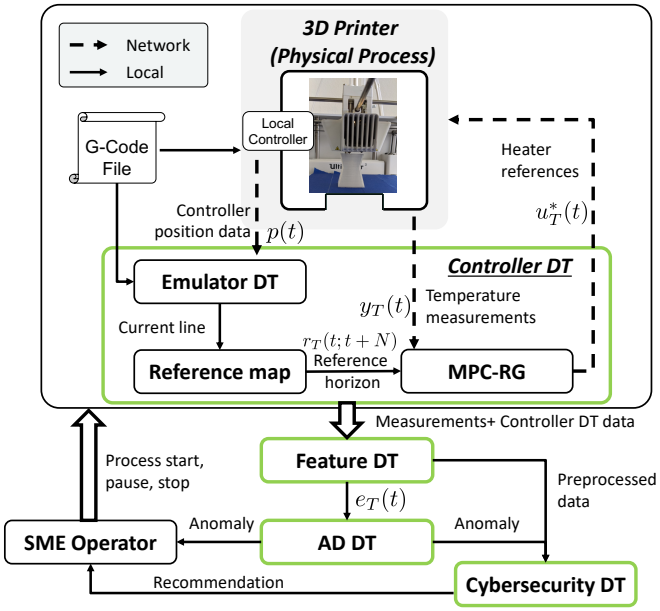

Fig. 5. Illustration of the DT architecture with the off-the-shelf 3D printer. Data communication over the network is shown with dashed lines and local data communication is shown with solid lines.

A. Controller DT over a Network

For our experimental demonstration, we focus on the heating system of an off-the-shelf fused filament fabrication (FFF) 3D printer. In FFF, a thermoplastic material is extruded onto a build bed via a numerically controlled extruder with a heated nozzle. A G-Code file is an input to the printer's local controller, and the local controller executes each line of G-Code in sequence to deposit material at each layer to create a 3D geometry in a bottom-up, layer-by-layer fashion. Thus, a physical process is operated by purely cyber inputs.

1) Motivation: Heating the deposited material within the desired temperature range is crucial for an extrusion process. The local controller includes a Proportional-Integral-Derivative (PID) loop that ensures robust tracking of a temperature reference $r_T(t)$ prescribed by the G-Code file, however, dynamic updates to the printing temperature are of interest for several reasons. Dynamic adjustment of printing temperatures is shown to greatly improve dimensional performance [48] as well as layer-to-layer material adhesion and part strength [49], [50]. To enable such applications of interest, we implement a network Controller DT that adjusts the printing temperature of the 3D printer based on a *reference map* that is designed by an engineer for a specific printing process. Run-time communication of the heater inputs over a network induces potential cyberattack vulnerabilities that may cause the failure of the printed part or the machine itself. For this purpose, we demonstrate how our framework enables DT-based cybersecurity solutions for the controller physical process in the context of expected anomalies.

2) Controller Implementation: Since we do not have direct access to the nozzle heaters in the printer, we model the closed-loop heating system (i.e., heaters controlled by the local controller) and develop a model predictive controller (MPC) scheme to prescribe heater references so that the system output $y_T(t)$ tracks a reference temperature $r_T(t)$. To implement a controller, the heating system is modeled as a discrete-time second order linear time invariant (LTI) system

$$\boldsymbol{x}(t+1) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}u_T(t) \tag{9a}$$
$$y_T(t) = \boldsymbol{C}\boldsymbol{x}(t), \tag{9b}$$

where the system matrices $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$, $\boldsymbol{B} \in \mathbb{R}^{2\times 1}$ are identified from the step response of the closed-loop heating system with $\boldsymbol{C} = [1 \quad 0]$. We define the control input limits as $\mathcal{U} = [160, 220]$ °C. Then, the goal of the MPC controller is to solve the following optimization problem in a receding horizon fashion

$$\min_{\boldsymbol{u}} \sum_{\tau=t}^{t+N-1} ||\boldsymbol{x}(\tau) - \boldsymbol{x}^r(\tau)||_Q^2 + ||u_T(\tau) - u_T^r(\tau)||_R^2 \tag{10a}$$
$$+ ||\boldsymbol{x}(t+N) - \boldsymbol{x}^r(t+N)||_P^2 \tag{10b}$$
$$\text{s.t.: } \boldsymbol{x}(\tau+1) = \boldsymbol{A}\boldsymbol{x}(\tau) + \boldsymbol{B}u_T(\tau) \tag{10c}$$
$$u_T(\tau) \in \mathcal{U}, \boldsymbol{x}(t) = \hat{\boldsymbol{x}}(t), \quad \tau = t, \ldots, t+N-1 \tag{10d}$$

where we have Q, R, P as positive definite controller gains, $||x||_Q^2 = x^T Q x$, $\boldsymbol{x}^r$ and u_T^r as the state and control input references, respectively, $\hat{\boldsymbol{x}}(t)$ as the current state estimate, N as the controller horizon, and $\boldsymbol{u} = \{u_T(t), u_T(t+1), \ldots, u_T(t+N-1)\}$. We use the solution of the corresponding Discrete-time Algebraic Ricatti Equation (DARE), i.e., the LQR gain, for defining the weight matrix P. We denote the optimal solution of (10) with $\boldsymbol{u}^*$. After the controller implements $u_T^*(t)$ on the physical system over the network, the optimization (10) is solved over an updated horizon with updated process data. We use a standard Kalman filter observer update to estimate the current state $\hat{\boldsymbol{x}}(t)$ and omit the formulation here for brevity.

3) Reference handling: As the formulation (10) suggests, the controller operates in the temporal domain. However, G-Code references executed on the printer are inherently spatial and event-based. To remedy this mismatch, we utilize an Emulator DT that emulates the printing process by analyzing the G-Code file. During run-time, the Emulator DT queries the position data of the four axes (e.g., x,y,z location of the extruder head, and the position of the extrusion (E) axis), $p(t) \in \mathbb{R}^4$, from the local controller. Then, the Emulator DT utilizes $p(t)$ to estimate the current line of G-Code executed by the printer. For our case study, we utilize a temperature reference that alternates between 205°C and 210°C every five layers in the printing process.

B. Attack and Anomaly Scenarios

We consider two attack scenarios and two anomaly scenarios in our case study.

1) Anomalies: Two types of anomalies are considered:

A1 The first anomaly is caused by the use of a cooling fan on the extruder head. The fan increases the airflow over the extruder nozzle, which reduces its temperature. The cooling effect is an exogenous disturbance that is unknown to the controller and causes an anomaly in the temperature measurements.

A2 The second anomaly is the degradation of the heating system performance. As the heating system is used over time, its components undergo thermal and mechanical wear which causes the system response to be slower (in terms of settling time) than expected for a given temperature reference $r_T(t)$. As this effect occurs gradually over a long time horizon (a matter of months of use), we instead simulate the degradation by updating the local controller gains to deliberately slow down the closed-loop response of the local heating system.

As shown in Fig. 5, an anomaly detection (AD) DT is implemented as a threshold-based limit-checking procedure on the temperature error $e_T(t) = r_T(t) - y_T(t)$. Thus, the AD-DT checks if the error is larger than a predefined threshold level $\beta_{AD} \in \mathbb{R}_+$, i.e., $|e_T(t)| > \beta_{AD}$. The value of β_{AD} is preset by a designer based on the expected system response characteristics (e.g., expected maximum temperature error, robustness margins, etc.).

2) Attacks: As the DT framework communicates with the printer only over a network, the measurement signals may be prone to attacks. Note that an attack on the measurement signal over the network is a vulnerability of the network communication and not the DTs themselves. Thus, the attack scenario in this case study is similar to network attacks studied extensively in the literature for controlled systems, see e.g., [28]. We utilize our DT framework in the context of the said network attacks to detect and differentiate them from expected anomalies. To simulate network attacks on the measurements, we consider two attack types as $y_T(t) + w(t)$, where $w(t)$ is the attack signal we implement on the measurement.

T1 Injection of a constant offset to the measurement signal, e.g., $w(t) = c_1$ for some $c_1 \in \mathbb{R}$.

T2 Injection of a temporally cyclic signal to the measurement signal, e.g., $w(t) = c_2 \sin(t)$ for some $c_2 \in \mathbb{R}$.

In this case study, we are focused on the attacks that compromise the temperature measurements $y_T(t)$. Attacks on the transmission of heater reference input ($u_T(t)^*$) from the Controller DT to the 3D printer are not considered. However, we note that various attack types may be implemented on the case study setup by considering corresponding measurements and signals from the system.

Remark 5. *Note that as mentioned in Remark 1, we do not differentiate attacks based on their malicious or non-malicious intent. Within the scope of this work, we are only interested in the output measurable effect of an attack on the system (see e.g., Fig. 1). Therefore, we present two attack scenarios with two different measurable output effects on the experimental system based on common network attack types. At the end of this section, we provide a further discussion on how different attack scenarios for various manufacturing resources from the literature can be implemented by our proposed framework.*

Although we pose the two attacks in this study as instances of common network attacks, they could represent the measurable output of many other attack types. For example, considering the examples given in Fig. 1, a compromised sensor on the physical system or upgrade errors causing false measurement readings due to miscalibration could result in similar output measurable effects.

C. Cybersecurity DT

Following the architecture illustrated in Fig. 3, the Cybersecurity DT is designed for abnormality detection and consistency checking. We present a proposed implementation for the Cybersecurity DT in this section, and refer the reader to the previous sections for discussions on how other solutions from the literature can be implemented as part of the proposed DT framework instead.

1) Detector DT: As previously stated, the temperature reference alternates between the two setpoints every five layers. We utilize two one-class support vector machines (OSVM) [42], [51], [52] to model the normal behavior of the process at the two setpoints, one for each setpoint. An OSVM utilizes training data that correspond to the same class, also named as *positive training samples* e.g., measurements under normal operation. Let us denote the training data as $D^+ = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{n_z}\}$, where $\boldsymbol{z}_i \in \mathcal{Z}$ denote individual measurements. Utilizing a mapping $\phi : \mathcal{Z} \to \mathcal{F}$, the OSVM trains its data boundaries with the following optimization

$$\boldsymbol{\alpha}^* = \min_{\boldsymbol{\alpha}} \{\boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha} \mid 0 \leq \alpha_i \leq \frac{1}{v n_z}, \textstyle\sum_i \alpha_i = 1\}, \quad (11)$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_{n_z}]$ is the decision variable, $v \in \mathbb{R}_+$ is a user-defined regularizer parameter, and $\boldsymbol{Q}[i,j] = k(\boldsymbol{z}_i, \boldsymbol{z}_k) = \phi(\boldsymbol{z}_i) \cdot \phi(\boldsymbol{z}_j)$ with $k(\boldsymbol{z}_i, \boldsymbol{z}_k)$ representing a kernel function, which is in turn given by the dot product $\phi(\boldsymbol{z}_i) \cdot \phi(\boldsymbol{z}_j)$. The optimal threshold value is evaluated as

$$\rho^* = \textstyle\sum_i \alpha_i k(\boldsymbol{z}_j, \boldsymbol{z}_i). \quad (12)$$

The decision function for one-class classification is given as

$$h(\boldsymbol{z}_*) = sgn(\textstyle\sum_i \alpha_i k(\boldsymbol{z}_i, \boldsymbol{z}_*) - \rho^*). \quad (13)$$

Furthermore, we denote the trained model of the OSVM as

$$\mathcal{B}(D^+) = \{\boldsymbol{z} \mid h(\boldsymbol{z}) \geq 0\}. \quad (14)$$

If we have a sample $\boldsymbol{z} \in \mathcal{B}(D^+)$, the Detector DT predicts that the sample is normal, and abnormal otherwise. Note that if we are training only on the measurement outputs $\boldsymbol{y}(t)$, we may utilize $\phi = \psi$, where the map ψ is given previously. For further details on the derivation of (11), see [40], [52]. For our implementation, we utilize the output measurements to train our OSVM, i.e., $\boldsymbol{z}_t = y_T(t)$.

To collect training data D^+, we run the process at the given setpoint temperatures (denoted with T_1^s and T_2^s) and with the

MPC providing heating references in closed-loop. We train two OSVMs on the collected data at two different setpoints, denoted with $D^+(T_1^s)$ and $D^+(T_2^s)$ to evaluate the two models as $\mathcal{B}_1 := \mathcal{B}(D^+(T_1^s))$ and $\mathcal{B}_2 := \mathcal{B}(D^+(T_2^s))$.

To deal with controlled transient behavior, we utilize the solution map ϕ of the linear system model (9). Since the Cybersecurity DT is provided with data from the Controller DT, the discrete-time index when the system is driven to a new setpoint is tracked as $t = n_{sp}$. Then, by querying the sequence of previous inputs $\boldsymbol{u} = \{u_T^*(t-1), u_T^*(t), \ldots, u_T^*(t-n_{sp})\}$ from the Feature DT, the Detector DT evaluates the projected state $\bar{\boldsymbol{x}}(t-n_{sp})$ by utilizing the state estimates $\hat{\boldsymbol{x}}$ (see (4)). Since the OSVM is trained on the output measurements, we further get the corresponding projected output measurement as $\bar{y}_T(t-n_{sp}) = \boldsymbol{C}\bar{\boldsymbol{x}}(t-n_{sp})$. If we have $\bar{y}_T(t-n_{sp}) \in \mathcal{B}_i$, where i denotes the corresponding previous setpoint, then the Detector DT predicts that the current state estimate $\hat{\boldsymbol{x}}(t)$ is normal and abnormal otherwise.

Remark 6. *The abnormality detection checks the condition $\bar{y}_T(t-n_{sp}) \in \mathcal{B}_i$. If the volume of $\mathcal{B}_i$ is too large (in a multidimensional sense), projections of certain attacked process measurements may still be within $\mathcal{B}_i$, resulting in false negatives. Additional models to refine the abnormality predictions of the Detector DT may be utilized to improve false negatives in such cases.*

2) Consistency DT: We present a consistency DT designed by utilizing expertise knowledge about the controlled physical process. By including the Controller DT in our framework, we have additional information about the expected system behavior under closed-loop control. Namely, the controller (10) provides near offset-free tracking under perfect model and state knowledge (see [53] for further details). Since we have inherent uncertainties in our model (9) as well as state estimation, we expect the controller to have a small steady-state tracking offset (in the normed sense), which we evaluate experimentally (an over-approximation of this offset is denoted with δ_1). Then we define a consistency metric

$$\xi_1(t) = c_1(y_T(t), r_T(t)) = |y_T(t) - r_T(t)|, \tag{15}$$

which provides us with the norm of the output measurement residual signal. For a consistent physical system, the residual should converge to a neighborhood of the empirically determined tracking offset, e.g., $\xi_1(t) \leq \delta_1$ as $t \to \infty$. However, monitoring the asymptotic response of the system is often not feasible or desirable. Let $\tau(t) \in \{0, 1\}$ denote a consistency metric request, such that we have $\tau(t) = 1$ if either the AD-DT or the Detector DT requests a consistency metric and $\tau(t) = 0$ otherwise. We define $\xi_2(t, t_0, t_f)$ as another consistency metric to count the number of requests in a given time interval $t \in [t_0, t_f]$. We use ξ_2 to define specifications that are robust to transient response in the state estimation, which often occur due to disturbances on the process and data communication. Using STL, the monitoring logic is given as

$$\pi_1 : \tau(t) \implies (\tau(t+1) \vee \Diamond_{[t+1,t+t_s]}(\xi_1(t) \leq \delta_1)), \tag{16a}$$
$$\pi_2 : \tau(t) \implies \Diamond_{[0,\beta]}(\neg\tau(t)), \tag{16b}$$
$$\pi_3 : \tau(t-1) \wedge \neg\tau(t) \implies \Box_{[0,\beta_2]}(\xi_2(t, 0, \beta_2) \leq 1), \tag{16c}$$
$$\pi_4 : \tau(t) \wedge (\xi_1(t) \geq 1) \implies \Diamond_{[t+1,t+t_s']}(\xi_1(t) \leq \delta_1), \tag{16d}$$

where (16a) denotes that whenever the consistency metric is requested, (i.e., $\tau(t) = 1 = \top$); either the consistency metric is requested again in the next time-step $(t+1)$, or the temperature error norm eventually converges to the tracking offset δ_1 within t_s time-steps. Similarly, (16b) requires that whenever the consistency metric is requested, i.e., the measurement is abnormal, the new measurements should be normal eventually in the next β time steps, where the value of β is determined empirically based on previous process data. Next, (16c) requires that whenever the output measurement turns normal (i.e., request is made in the previous time step but not the current one), it should stay that way for the next β_2 time-steps based on the ξ_2 consistency metric (i.e., there can be at most one consistency metric request in the given time interval). Similar metrics may be developed for various applications of interest. Lastly, (16d) denotes that if there is a large deviation in the signal at the time of consistency metric request, the signal should converge to the δ_1 limit within t_s' timesteps. Note that π_3 and π_4 may be merged for a succinct representation, but we utilize two separate specifications here for clarity. Furthermore, statistical metrics such as ARL may be utilized to tune the parameters for the STL monitoring parameters, similar to the statistical analysis provided in [29]. We instead utilize empirical data collected on our experimental setup to tune these parameters, given that our process is non-stationary in the sense that the controlled system includes a reference tracking MPC, and there are reference changes inducing transient dynamics.

We denote the STL specification

$$\pi = \pi_1 \wedge \pi_2 \wedge \pi_3 \wedge \pi_4, \tag{17}$$

which is satisfied (`SAT`) if and only if all the propositions in (16) are satisfied (i.e., they are $\top$). The Consistency DT monitors the process data to check if the specification (17) is `SAT`. The monitoring process may be performed in runtime with robust satisfaction monitoring techniques [54]. Here we instead utilize a retrospective analysis on the collected process data stored in the Cybersecurity DT to perform the monitoring task. We expect a consistent process to have the output measurement stable under the closed-loop control implemented by the Controller DT, resulting in `SAT`. When the measurement signal is compromised, the controller will not be able to track the desired reference, which causes large tracking errors or instability on the temperature signal that result in (17) to be unsatisfied (`UNSAT`).

We determine the value of the parameter t_s' based on the expected system performance as a constant parameter during the process. However, the parameter t_s should be adjusted based on the initial condition of the mismatch between $y_T(t)$ and $r_T(t)$ dynamically, since the system dynamics have a non-trivial time constant. To evaluate the parameter t_s in (16a)

we utilize a user-defined parameter mapping for t_s, which is determined based on the analysis of previous anomalous data by an expert.

D. Implementation Details

We use an Ultimaker 3[1] FFF printer with Polylactic Acid (PLA) material for our case study. The printer has a network application programming interface (API) that allows for monitoring of extruder temperatures $y_T(t)$, stepper motor counts on the controller $p(t)$, and extruder heater input updates $u_T(t)$. The Controller DT is implemented on a personal computer using python. For reliable network communication between the Controller DT and the printer in run-time, a sampling time of 0.5 seconds is used in the case study. In each 0.5 second cycle, the Controller DT execution sequence is repeated and a new heater reference $u_T^*(t)$ is sent to the printer over the network. The linear model in (9) is adjusted for actuator delays, which were approximated to be 10 time steps for the implementation of the control law using (10).

We collect experimental time-series data for (i) the nominal system without the Controller DT running, (ii) the controlled system with the Controller DT without attacks or anomalies, (iii) the controlled system with anomaly cases (***A1*** and ***A2***), and (iv) the controlled system with sensor attack cases (***T1***, ***T2***). For experimental implementation purposes, we replay the collected time-series data to the Cybersecurity DT for asynchronous data analysis in an offline fashion. This approach is taken to avoid any computational errors in our results due to possible run-time execution issues and to illustrate how various DTs in the framework may operate at different time scales. It is important to note that synchronous implementations of the Cybersecurity DT would perform comparably in practice to our approach since we do not change how the Cybersecurity DT interacts with the other DTs in the framework in our experimental approach. The experimental data used in this case study is available in the supplementary materials.

E. Results

In this section, we present experimental study results from the implementation of the proposed framework. Figures in this section are the results of multiple runs of the printing process for various anomalies and attacks. Therefore the time axes on the processes do not illustrate times from a single printing process but rather show the time scale of the dynamic behavior.

1) Detector DT performance: We use nominal process data when the Controller DT is running to train the OSVMs for the two temperature setpoints $T_1^s = 205°\text{C}$ and $T_2^s = 210°\text{C}$. We use radial basis function kernels for our initial OSVM training. The transient analysis in (4) is sensitive to model mismatches, which results in false-positives on normal data traces. Thus, we analyze the performance of the OSVMs on the nominal process measurement and transient responses to improve their robustness by perturbing their data boundaries. We perturb the boundaries based on the performance of the

[1]Certain commercial instruments and materials are identified to specify the experimental study adequately. This does not imply endorsement by NIST or that the instruments and materials are the best available for the purpose.

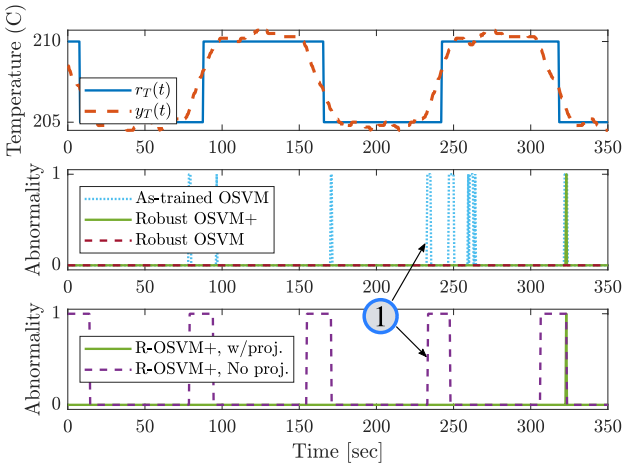

Fig. 6. Illustration of the Detector DT abnormality detection during normal operation using the robust bounds.

Detector DT on the historical data to mitigate false positives (e.g., include historical data points that are known to be normal but predicted abnormal by the Detector DT in the training set D). This procedure reduces the false positive rates of the Detector DT at the expense of reduced sensitivity (see Remark 6). The perturbation procedure is designed such that the resulting boundaries of the *Robust OSVM* are an over-approximation of the original *as-trained OSVM*.

To remedy the reduced sensitivity of the Detector DT with the Robust OSVM due to the perturbation procedure, we utilize additional OSVMs to provide a solution to the scenario outlined in Remark 6. We train two new additional OSVMs (one for each setpoint) on the features of the Robust OSVM solutions for normal data points. The output of the Detector DT with these new additional OSVMs is denoted as *Robust OSVM+* in Fig. 6 and the indication $+$ is used in subsequent figures for comparisons to the two different Robust OSVM solutions. To train the Robust OSVM+, we extract three features from the Robust OSVM solutions on the normal dataset. Specifically, we train the Robust OSVM+ on the

- Root-mean-squared error of the estimation (4)
- Time elapsed since the last setpoint
- The absolute estimation error $|\bar{y}_T - y_T|$,

of the Robust OSVM solutions on the normal dataset. We pass the datapoints that are predicted to be normal by the Robust OSVM through the Robust OSVM+ for additional analysis and to improve the Detector DT performance.

Figure 6 illustrates the performance of the Detector DT with the updated robust data boundaries on normal process data. The top plot illustrates the temperature reading and the reference temperature for the process. The middle plot shows the comparison of the predictions using the As-trained OSVM, Robust OSVM, and Robust OSVM+. We observe that the As-trained OSVM results in a high rate of false positives in the middle plot, annotated with 1 on the figure. The Robust OSVM+ solution in the middle plot has slightly increased sensitivity that still rejects most of the As-trained OSVM false positives. We show how the increased sensitivity,

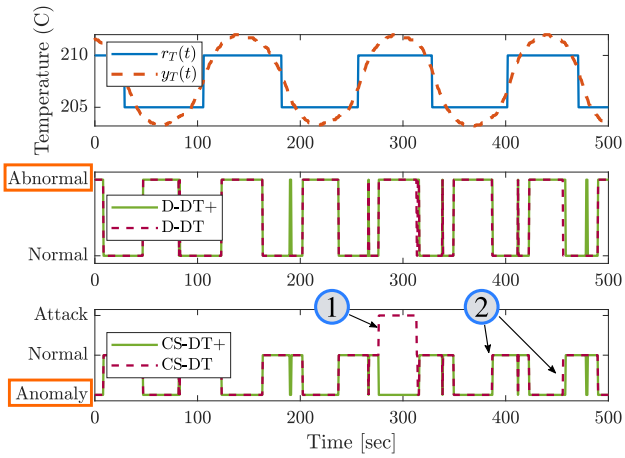

Fig. 7. Illustration of the Detector DT abnormality detection and the Cybersecurity DT attack detection during the anomaly ***A2***. **Top** plot shows the temperature response of the process and the reference temperatures. **Middle** plot shows the Detector DT outputs for the Robust OSVM and Robust OSVM+ procedures with D-DT and D-DT+ respectively. **Bottom** plot shows the Cybersecurity DT outputs when the Robust OSVM and Robust OSVM+ procedures are utilized with CS-DT and CS-DT+ respectively. The rectangles around Abnormal and Anomaly indicate the ground truth for the signals.

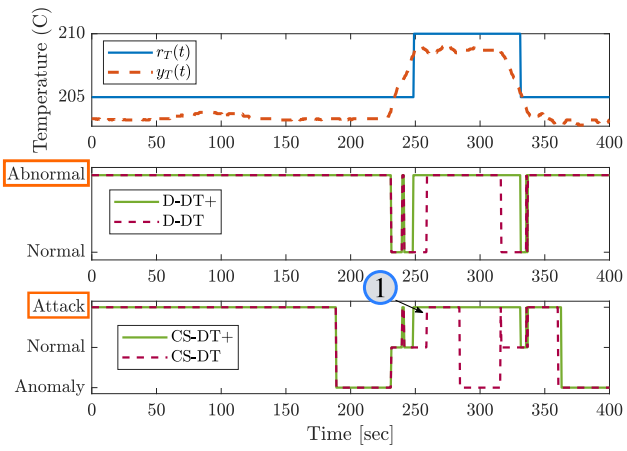

Fig. 8. Illustration of the Detector DT abnormality detection and the Cybersecurity DT attack detection during the attack ***T1***. **Top** plot shows the temperature response of the process and the reference temperatures. **Middle** plot shows the Detector DT outputs for the Robust OSVM and Robust OSVM+ procedures with D-DT and D-DT+ respectively. **Bottom** plot shows the Cybersecurity DT outputs when the Robust OSVM and Robust OSVM+ procedures are utilized with CS-DT and CS-DT+ respectively. The rectangles around Abnormal and Attack indicate the ground truth for the signals.

when compared to the Robust OSVM, provides better attack detection performance in the following results.

On the bottom plot of Fig. 6 we illustrate the importance of the projection method given in (4) utilizing the Robust OSVM+ (R-OSVM+) procedure. The output without the projection (R-OSVM+, No proj.) has excessive false positives, annotated with 1 on the figure, since it predicts all the transient responses as abnormal datapoints. It is important to note that our proposed method for abnormality detection during controlled transient response works with minimal false positives in our experimental setup. The data trace (R-OSVM+, w/proj.) correctly identifies the transient behavior of the process as normal behavior and has minimal false positives (e.g., the one spike at time ≈ 225 seconds on the bottom plot of Fig. 6).

2) Cybersecurity DT performance: Here we analyze the performance of the Cybersecurity DT for the attack and anomaly cases considered in our case study. All the figures have rectangles around certain axis labels indicating the ground truth for the signal illustrated in the figure. Additionally, we provide numbered annotations on the figures to highlight some discussion points that we provide in the text.

Figure 7 illustrates the Detector DT and Cybersecurity DT output for the anomaly case of ***A2***. On the top plot, we see that the system response is altered due to the change in the local controller parameters. Specifically, the system response is slower than what is anticipated by the model in the controller. Consequently, the temperature signal significantly over- and undershoots the reference signal as the controller has a large model mismatch, which is illustrated by the process response on the top plot of Fig. 7. The Detector DT predicts abnormalities throughout the process and requests consistency metrics from the Consistency DT. In the middle plot we see the effect of the Robust OSVM+ versus the Robust OSVM, where the predicted abnormalities are slightly different for the two cases. Due to the anomaly, the datapoints throughout Fig. 7 are abnormal and anomalous, which is indicated by the rectangles in the axis labels. The detector DT identifies the abnormalities as shown in the middle plot, during the setpoints and partially during the transient response between the setpoints. The transient response of the system is similar to the normal case, which results in the Detector DT predicting normal outputs during parts of the transient response.

Since the process is not attacked, we see that the temperature output is still bounded around the reference trajectory and thus the predictions of the Cybersecurity DT are *normal* where the temperature signal begins to converge towards the reference. On the bottom plot of Fig. 7, we see that the Cybersecurity DT correctly identifies the anomaly in the signal for the abnormal predictions identified by the Detector DT. Additionally, we see that the Robust OSVM, due to its reduced sensitivity in abnormality detection, results in a false postive of the Cybersecurity DT which can be seen on the bottom plot of the Fig. 7, annotated with 1. In comparison, the Robust OSVM+ has no false positives (e.g., all the predictions are either anomalous or normal). Similar to the case illustrated in Fig. 7, the Cybersecurity DT was able to correctly detect the anomaly ***A1*** for all the experimental data we have collected.

Figure 8 illustrates the Detector DT and Cybersecurity DT output for the attack case ***T1***. Due to the attack on the system, the controller stabilizes the temperature outputs at an offset away from the desired setpoints. The middle plot shows the abnormality predictions of the two OSVM procedures. The Robust OSVM+ has better abnormality prediction (shown with D-DT+) when compared to the Robust OSVM (shown with D-DT). Using the consistency measures and the STL monitoring, the Consistency DT analyzes the measurements and identifies the offset in the signal. On the bottom plot of Fig. 8 we see that the Cybersecurity DT identifies attacks on the inconsistent measurements from the process. During transients in the process, the signal behaves consistently with

the nominal transient behavior, which causes the Cybersecurity DT to predict normal measurements. The annotation 1 on the bottom plot highlights the attack prediction of the CS-DT, which is later and more inconsistent when compared to the CS-DT+ outputs that use the Robust-OSVM+. We clearly see that the Robust OSVM+ in the Detector DT improves the abnormality detection and consequently attack detection performance of the Cybersecurity DT.

Figure 9 illustrates the Detector DT and Cybersecurity DT output for the attack case ***T2***. Due to the attack on the system, the controller is not able to stabilize the system to any reference temperature which can be observed on the top plot. Thus, the temperature measurements fluctuate irregularly causing the Detector DT to detect abnormality most of the time, shown in the middle plot. Using the consistency measures and the STL monitoring, the Consistency DT analyzes the measurements due to the requests from the AD-DT and the Detector DT. On the bottom plot of Fig. 9 we see that the Cybersecurity DT identifies attacks on the inconsistent measurements from the process. It is important to note that the Cybersecurity DT finds normal or anomalous measurements in the data stream at times where the measurement signal is similar to the transient response of the nominal process or the abnormal signals of the anomalous process. Annotation 1 on the bottom plot shows the missed positive by the Robust OSVM when compared to the Robust OSVM+. While the effect of Robust OSVM+ is attenuated in this case, we still see that when compared to the Robust OSVM, the Robust OSVM+ provides slightly better attack detection performance. The anomalous predictions between $100-150$ seconds and $170-210$ seconds exhibit similar behavior to the anomaly case ***A2***, which result in the Cybersecurity DT predicting anomalies instead of attack signals in those intervals. As mentioned earlier, the recommendation of an attack on the system is shared with an attack classifier (either to an SME or to additional data analysis DT) for further analysis.

3) Discussion: The results in this section show that the Cybersecurity DT identifies inconsistencies throughout the signal, which predicts that the signal is most likely compromised, i.e., attacked. Furthermore, when used for anomalous signals and attack signals interchangeably, the Cybersecurity DT was able to correctly analyze and identify an attack on the system without any need for parameter adjustment or model re-training. Due to the way we have implemented the STL specifications for consistency monitoring, we have observed that some attacked signals are partially missed or identified as anomalous especially around setpoint changes. This is an important tradeoff in the framework design that we favored to provide robustness when the process is in nominal condition (no anomalies or attacks). AM processes often take multiple hours to manufacture a 3D object. Therefore, false alarms that would result in stopping a process prematurely could be costly (e.g., time and material lost due to a false alarm at the end of a multi-hour print job). Consequently, our design is aimed toward reducing false alarms on the physical process. Various extensions that allow for higher false alarm rates may be utilized for processes where the cost of missing an attack is much higher than prematurely stopping the process.

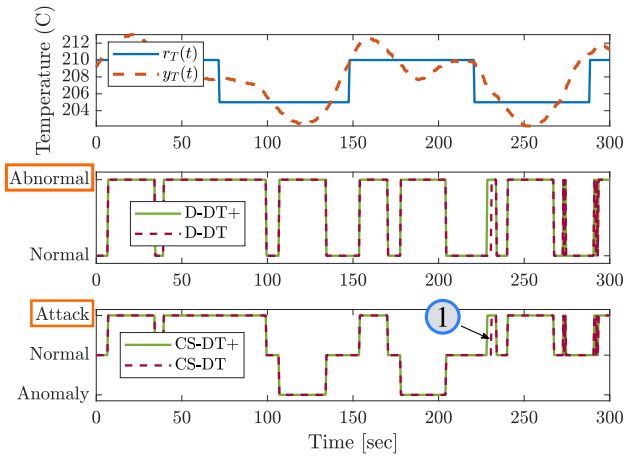

Fig. 9. Illustration of the Detector DT abnormality detection and the Cybersecurity DT attack detection during the attack ***T2***. **Top** plot shows the temperature response of the process and the reference temperatures. **Middle** plot shows the Detector DT outputs for the Robust OSVM and Robust OSVM+ procedures with D-DT and D-DT+ respectively. **Bottom** plot shows the Cybersecurity DT outputs when the Robust OSVM and Robust OSVM+ procedures are utilized with CS-DT and CS-DT+ respectively. The rectangles around Abnormal and Attack indicate the ground truth for the signals.

We demonstrated a possible implementation for the Cybersecurity DT in this case study, utilizing OSVM detectors in conjunction with STL consistency checking. As previously discussed, several other methodologies from the literature may be utilized instead within our DT framework, thanks to its modular and flexible design. Comparing our framework to others in the literature, we note that many of the existing work focus on either anomaly or attack detection. Thus, implemented alone, our methodology would outperform such methods in detecting and distinguishing attacks versus anomalies.

To compare our method to recent work such as [29], where equipment faults are also considered, we make remarks throughout the sections to illustrate how their method can be implemented with our framework. Also, in our proposed framework, we do not limit the dynamical systems or the applicable controllers to derive statistical conditions as presented in [29], but rather outline a framework of DTs that can be developed flexibly based on the specific need. We demonstrate an optimization-based controller with dynamically changing references to illustrate a very general case for the generalizability and flexibility of our proposed DT framework, when compared to the existing literature.

An important observation about the attack detection results indicate that the detection performance may be improved by considering a latched process detection. In our presented work, all detections are conducted by comparing the datapoints to the expected normal boundaries; however, by incorporating the state of the process as anomalous, normal, or attacked, further methods can be developed to instead look at the transition from one state to another. Such detection methods may require additional modeling and data analysis, and would be an extension of the proposed framework in this work.

F. Outlook for Additional Application Scenarios

In this case study, we demonstrated the utility of our framework on an off-the-shelf 3D printer. Our framework is capable of accommodating various CPMS with minimal modifications. In this section, we briefly discuss a few other examples from the literature and how our framework enables tools to implement them in practice.

We note that our detection and consistency DTs require a certain level of knowledge (both physics-based and SME knowledge) to design the given metrics and DTs, and to have necessary sensors to measure the effect of an attack. However, this is not a limitation in numerous practical cases such as the broad class of physics-based detection methods. Additionally, consistency features may be learned from past data using data-driven classification and pattern matching methods, reducing the need for SME knowledge in designing a consistency DT. As output measurable attacks are of interest in this study, we assume that necessary sensors are available in the given CPMS. Next we discuss recent attack detection examples from the literature and how they can be implemented using our framework.

In [24], product-oriented attacks for machining processes are studied. In-situ monitoring of process variables including side-channel measurements is utilized for monitoring and attack detection. In our framework, the measurement systems can be implemented as part of the physical process and/or via the Controller DT. The measurement data is processed by the Feature DT to evaluate the metrics mentioned in [24] (e.g., Material Removal Rate, cutting time, etc.), logged in a database, and shared with the Cybersecurity DT. The Cybersecurity DT may be implemented with the attack detection logic presented in [24], where the Consistency DT can implement STL logic to monitor the changes in these parameters to be within limits specified by an SME.

In [26], data-driven methods for detecting cyber-physical attacks are proposed and demonstrated on an AM process and a machining process. The measurement data streams can be processed by the Feature DT to prepare necessary features to analyze by the Cybersecurity DT. The data-driven methods proposed in [26] can be implemented as part of the Detector DT, similar to the OSVM application we have demonstrated in our work. Using the models from [26], decision logic flows similar to those given in Fig. 4 can be used for cyber-attack detection. From the specific case studies in [26], the microcontroller used for collecting images on the 3D printer streams images to the framework via the Feature DT, where the images are logged to a database, and feature signals such as mean, standard deviation, and magnitude of pixel values are evaluated by the Feature DT and shared with the Cybersecurity DT. Then, the data-driven models are utilized to detect cyber-attacks and alert an SME, just as given in Fig. 2. However, methods to differentiate anomalies in the context of attacks and dealing with advanced closed-loop controllers with set point changes and transient behavior has not been discussed in [24], [26].

Many of the other works from the literature utilize side-channel measurements to detect cyber-attacks. These methods can be implemented within our framework using appropriate Detector DTs to identify signal signature changes and Consistency DTs to differentiate attacks (e.g., [22], [25]). Additionally, the digital signature matching process in [23] can be implemented in our Cybersecurity DT via the Detector DT and the Consistency DT with the remaining data streams implemented similarly to the examples given above. The data processing and attack detection methods in [29] can be implemented with Feature and Detection DTs. Additionally, the differentiation method and the process controllers considered in [29] may be implemented as part of the Consistency and Controller DTs. On top of being able to accommodate the solutions from [29], our framework further enables tools to deal with a large class of attack types and the use of advanced control methods with transient behavior. Similarly, SPC methods [46] may be utilized for consistency metrics within our DT framework. Just as existing AD methods can be integrated into Feature and AD DTs in the framework. SPC methods and control charts can be integrated into the Detector DT to find traces of statistically significant variations, which then may be further analyzed by the consistency DT. We note that many of the threshold-based SPC rules, e.g., the western electric (WE) rules, may be implemented using the proposed consistency DT in a scalable manner. A detailed study of implementing the full set of SPC tools within our framework is beyond our scope and subject for future work.

It is important to note that our framework is able to accommodate all these solutions from the literature while enabling tools for additional functionality such as anomaly detection and classification of anomalies from cyber-attacks in a practical setting as well as dealing with the transient response of the controlled physical process under set point changes due to a controller.

VI. Conclusion

We have presented a DT framework for cyberattack detection on CPMS in the context of closed-loop controllers and anomalies in the physical process. We demonstrated our approach on an off-the-shelf 3D printer on which we implemented a novel network controller. The proposed Cybersecurity DT is able to detect attacks and anomalies on the system while the process is controlled to switching setpoints. Our approach is platform agnostic and modular thanks to its DT-centric design. The components of the framework can be extended for multiple resources in a manufacturing system and aggregated into a system-level DT framework. Extensions of the presented approach for an AM fleet scenario [55] with multiple machines is of interest for future work. We note that in a plant floor with multiple machines, a designer could potentially utilize similar consistency and detection structures for similar machines, if possible, or design individual cybersecurity DTs for each machine, based on the application of interest. Developing methods for attack detection across multiple machines is a promising future direction.

We have designed and tested our approach for isolated attacks and anomalies. However, there may be practical cases where anomalies and attacks are present in the system simultaneously. Investigation of such cases is subject to future

work, especially in the context of controlled transient behavior. Additionally, testing the performance of a latched process approach (i.e., detecting changes from a current detection state to another instead of comparing all to the normal detection state) as identified in the discussion section is of interest. An important contribution of this work is the investigation of the abnormality detection under closed-loop control on the process. Further developments in detecting abnormalities during closed-loop transient behavior could improve CPMS performance and the abnormality detection performance of DTs. Additionally, we believe the presented OSVM based abnormality detection approach can be extended as a stand-alone AD approach for CPMS by defining the proper data labels for training and tuning the sensitivity of the detector for the specific use case (e.g., using a receiver operating characteristics curve). Lastly, studying attacks with no output measurable effects are of interest for future work.

ACKNOWLEDGMENT

The research for this work was conducted while Efe C. Balta was with the University of Michigan.

REFERENCES

[1] J. Davis, T. Edgar, J. Porter, J. Bernaden, and M. Sarli, "Smart manufacturing, manufacturing intelligence and demand-dynamic performance," *Computers & Chemical Engineering*, vol. 47, pp. 145–156, 2012.

[2] J. Moyne, Y. Qamsane, E. C. Balta, I. Kovalenko, J. Faris, K. Barton, and D. M. Tilbury, "A requirements driven digital twin framework: Specification and opportunities," *IEEE Access*, vol. 8, pp. 107 781–107 801, 2020.

[3] H. S. Kang, J. Y. Lee, S. Choi, H. Kim, J. H. Park, J. Y. Son, B. H. Kim, and S. Do Noh, "Smart manufacturing: Past research, present findings, and future directions," *International journal of precision engineering and manufacturing-green technology*, vol. 3, no. 1, pp. 111–128, 2016.

[4] F. Tao and Q. Qi, "New IT driven service-oriented smart manufacturing: framework and characteristics," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 1, pp. 81–91, 2017.

[5] C. Liu, P. Jiang, and W. Jiang, "Web-based digital twin modeling and remote control of cyber-physical production systems," *Robotics and Computer-Integrated Manufacturing*, vol. 64, p. 101956, 2020.

[6] Z. Yu, J. Ouyang, S. Li, and X. Peng, "Formal modeling and control of cyber-physical manufacturing systems," *Advances in Mechanical Engineering*, vol. 9, no. 10, p. 1687814017725472, 2017.

[7] F. Lopez, M. Saez, Y. Shao, E. C. Balta, J. Moyne, Z. M. Mao, K. Barton, and D. Tilbury, "Categorization of anomalies in smart manufacturing systems to support the selection of detection mechanisms," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 1885–1892, 2017.

[8] T. Yucelen, W. M. Haddad, and E. M. Feron, "Adaptive control architectures for mitigating sensor attacks in cyber-physical systems," *Cyber-Physical Systems*, vol. 2, no. 1-4, pp. 24–52, 2016.

[9] S. Belikovetsky, M. Yampolskiy, J. Toh, J. Gatlin, and Y. Elovici, "dr0wned–cyber-physical attack with additive manufacturing," in *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, 2017.

[10] R. Rosen, G. Von Wichert, G. Lo, and K. D. Bettenhausen, "About the importance of autonomy and digital twins for the future of manufacturing," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 567–572, 2015.

[11] F. Tao, H. Zhang, A. Liu, and A. Y. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, 2018.

[12] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in *Transdisciplinary perspectives on complex systems*. Springer, 2017, pp. 85–113.

[13] E. Negri, L. Fumagalli, and M. Macchi, "A review of the roles of digital twin in CPS-based production systems," *Procedia Manufacturing*, vol. 11, pp. 939–948, 2017.

[14] Y. Qamsane, J. Moyne, M. Toothman, I. Kovalenko, E. C. Balta, J. Faris, D. M. Tilbury, and K. Barton, "A methodology to develop and implement digital twin solutions for manufacturing systems," *IEEE Access*, vol. 9, pp. 44 247–44 265, 2021.

[15] G. Shao *et al.*, "Use case scenarios for digital twin implementation based on ISO 23247," *National Institute of Standards: Gaithersburg, MD, USA*, 2021.

[16] E. C. Balta, D. M. Tilbury, and K. Barton, "A digital twin framework for performance monitoring and anomaly detection in fused deposition modeling," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2019, pp. 823–829.

[17] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, "Guide to industrial control systems (ICS) security-NIST. SP. 800-82r2," *NIST, US Department of Commerce, Gaithersburg, Maryland*, pp. 1–247, 2015.

[18] L. F. Combita, A. A. Cardenas, and N. Quijano, "Mitigating sensor attacks against industrial control systems," *IEEE Access*, vol. 7, pp. 92 444–92 455, 2019.

[19] Z. Wang, F. Harirchi, D. Anand, C. Y. Tang, J. Moyne, and D. Tilbury, "Conflict-driven hybrid observer-based anomaly detection," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 5793–5800.

[20] F. Pasqualetti, F. Dörfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *IEEE transactions on automatic control*, vol. 58, no. 11, pp. 2715–2729, 2013.

[21] H. Song, P. Shi, C.-C. Lim, W.-A. Zhang, and L. Yu, "Attack and estimator design for multi-sensor systems with undetectable adversary," *Automatica*, vol. 109, p. 108545, 2019.

[22] S. Belikovetsky, Y. A. Solewicz, M. Yampolskiy, J. Toh, and Y. Elovici, "Digital audio signature for 3D printing integrity," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1127–1141, 2018.

[23] S.-Y. Yu, A. V. Malawade, S. R. Chhetri, and M. A. Al Faruque, "Sabotage attack detection for additive manufacturing systems," *IEEE Access*, vol. 8, pp. 27 218–27 231, 2020.

[24] M. S. Shafae, L. J. Wells, and G. T. Purdy, "Defending against product-oriented cyber-physical attacks on machining systems," *The International Journal of Advanced Manufacturing Technology*, vol. 105, no. 9, pp. 3829–3850, 2019.

[25] C. Liu, C. Kan, and W. Tian, "An online side channel monitoring approach for cyber-physical attack detection of additive manufacturing," in *International Manufacturing Science and Engineering Conference*, vol. 84263. American Society of Mechanical Engineers, 2020, p. V002T07A016.

[26] M. Wu, Z. Song, and Y. B. Moon, "Detecting cyber-physical attacks in cybermanufacturing systems with machine learning methods," *Journal of intelligent manufacturing*, vol. 30, no. 3, pp. 1111–1123, 2019.

[27] P. Mahesh, A. Tiwari, C. Jin, P. R. Kumar, A. N. Reddy, S. T. Bukkapatanam, N. Gupta, and R. Karri, "A survey of cybersecurity of digital manufacturing," *Proceedings of the IEEE*, vol. 109, no. 4, pp. 495–516, 2020.

[28] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell, "A survey of physics-based attack detection in cyber-physical systems," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[29] D. Li, N. Gebraeel, and K. Paynabar, "Detection and differentiation of replay attack and equipment faults in scada systems," *IEEE Transactions on Automation Science and Engineering*, 2020.

[30] A. Padovano, F. Longo, L. Nicoletti, and G. Mirabelli, "A digital twin based service oriented application for a 4.0 knowledge navigation in the smart factory," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 631–636, 2018.

[31] W. A. Jansen, T. Winograd, and K. Scarfone, "Guidelines on active content and mobile code: Version 2," in *National Institute of Standards and Technology (US)*, no. NIST Special Publication 800-28, Version 2. National Institute of Standards and Technology (US), 2008.

[32] R. Isermann, "Model-based fault-detection and diagnosis–status and applications," *Annual Reviews in control*, vol. 29, no. 1, pp. 71–85, 2005.

[33] D. Jung and C. Sundström, "A combined data-driven and model-based residual selection algorithm for fault detection and isolation," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 2, pp. 616–630, 2017.

[34] Y. Qamsane, C.-Y. Chen, E. C. Balta, B.-C. Kao, S. Mohan, J. Moyne, D. Tilbury, and K. Barton, "A unified digital twin framework for real-time monitoring and evaluation of smart manufacturing systems," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2019, pp. 1394–1401.

[35] J. Moyne, E. Del Castillo, and A. M. Hurwitz, *Run-to-run control in semiconductor manufacturing*. CRC press, 2000.

[36] J. Wang, L. Zhang, L. Duan, and R. X. Gao, "A new paradigm of cloud-based predictive maintenance for intelligent manufacturing," *Journal of Intelligent Manufacturing*, vol. 28, no. 5, pp. 1125–1137, 2017.

[37] Z. Song, A. Skuric, and K. Ji, "A recursive watermark method for hard real-time industrial control system cyber-resilience enhancement," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 2, pp. 1030–1043, 2020.

[38] Z. Tang, M. Kuijper, M. S. Chong, I. Mareels, and C. Leckie, "Linear system security—detection and correction of adversarial sensor attacks in the noise-free case," *Automatica*, vol. 101, pp. 53–59, 2019.

[39] E. C. Balta, Y. Lin, K. Barton, D. M. Tilbury, and Z. M. Mao, "Production as a service: A digital manufacturing framework for optimizing utilization," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1483–1493, Oct 2018.

[40] J. Zhang, K.-K. Ma, M.-H. Er, and V. Chong, "Tumor segmentation from magnetic resonance imaging by learning via one-class support vector machine," *International Workshop on Advanced Image Technology (IWAIT '04)*, pp. 207–211, 2004.

[41] J. Mũnoz-Marí, F. Bovolo, L. Gómez-Chova, L. Bruzzone, and G. Camp-Valls, "Semisupervised one-class support vector machines for classification of remote sensing data," *IEEE transactions on geoscience and remote sensing*, vol. 48, no. 8, pp. 3188–3197, 2010.

[42] V. Gómez-Verdejo, J. Arenas-García, M. Lázaro-Gredilla, and Á. Navia-Vázquez, "Adaptive one-class support vector machine," *IEEE Transactions on Signal Processing*, vol. 59, no. 6, pp. 2975–2981, 2011.

[43] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 665–674.

[44] B. Lindemann, F. Fesenmayr, N. Jazdi, and M. Weyrich, "Anomaly detection in discrete manufacturing using self-learning approaches," *Procedia CIRP*, vol. 79, pp. 313–318, 2019.

[45] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International Conference on Learning Representations*, 2018.

[46] J. S. Oakland, *Statistical process control*. Routledge, 2007.

[47] A. Donzé, T. Ferrere, and O. Maler, "Efficient robust monitoring for STL," in *International Conference on Computer Aided Verification*. Springer, 2013, pp. 264–279.

[48] D. S. Ertay, A. Yuen, and Y. Altintas, "Synchronized material deposition rate control with path velocity on fused filament fabrication machines," *Additive Manufacturing*, vol. 19, pp. 205–213, 2018.

[49] J. E. Seppala, S. H. Han, K. E. Hillgartner, C. S. Davis, and K. B. Migler, "Weld formation during material extrusion additive manufacturing," *Soft matter*, vol. 13, no. 38, pp. 6761–6769, 2017.

[50] F. Wang, S. Fathizadan, F. Ju, K. Rowe, and N. Hofmann, "Print surface thermal modeling and layer time control for large-scale additive manufacturing," *IEEE Transactions on Automation Science and Engineering*, 2020.

[51] S. Yin, X. Zhu, and C. Jing, "Fault detection based on a robust one class support vector machine," *Neurocomputing*, vol. 145, pp. 263–268, 2014.

[52] Y.-S. Choi, "Least squares one-class support vector machine," *Pattern Recognition Letters*, vol. 30, no. 13, pp. 1236–1240, 2009.

[53] G. Pannocchia and A. Bemporad, "Combined design of disturbance model and observer for offset-free model predictive control," *IEEE Transactions on Automatic Control*, vol. 52, no. 6, pp. 1048–1053, 2007.

[54] J. V. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia, "Robust online monitoring of signal temporal logic," *Formal Methods in System Design*, vol. 51, no. 1, pp. 5–30, 2017.

[55] E. C. Balta, D. M. Tilbury, and K. Barton, "A centralized framework for system-level control and management of additive manufacturing fleets," in *Automation Science and Engineering (CASE), 2018 14th IEEE Conference on*. IEEE, 2018.