HSU TwinFlow: A Living Benchmark for Evaluating Anomaly Detection and Diagnosis Methods in Cyber-Physical Production Systems Using Digital-Twin-Generated Data

Nemanja Hranisavljevic
Helmut Schmidt University, Hamburg, Germany

Alexander Diedrich
Helmut Schmidt University, Hamburg, Germany

Lukas Moddemann
Helmut Schmidt University, Hamburg, Germany

Domenic Schaeffer
Digital Twin Factory GmbH, Bad Oeynhausen, Germany

Frank Marek
Novatio Solutions GmbH, Monheim, Germany

Ingo Pill
Institute of Software Engineering and Artificial Intelligence, TU Graz, Graz, Austria

Oliver Niggemann
Helmut Schmidt University, Hamburg, Germany

Abstract

We present a living (evolving) benchmark for evaluating data-driven methods for fault-related tasks, including fault (anomaly) detection and diagnosis, based on datasets generated by a realistic digital twin of a cyber-physical production system (CPPS). The digital twin is designed to capture the structural layout, operational behavior, and material flow of a modern production facility in a realistic manner. Scenario definitions, represented as sequences of high-level events, are used to generate labeled model-training and evaluation data from carefully designed fault scenarios. In addition, the datasets include contextual data, such as scenario metadata, product configurations, material-flow information, together with system knowledge such as the component hierarchy, component-type information, and material-flow graph. The proposed benchmark therefore provides a practical basis for method development and qualitative evaluation beyond simplified or isolated test setups. We provide initial benchmark results by evaluating two representative methods for each of the two considered tasks: anomaly detection and fault diagnosis.

2012 ACM Subject Classification Computing methodologies → Artificial intelligence; Computer systems organization → Embedded and cyber-physical systems; Computer systems organization → Dependable and fault-tolerant systems and networks; Computing methodologies → Modeling and simulation; Computing methodologies → Machine learning

Keywords and phrases Benchmark, Digital Twin, Manufacturing, Production Systems, Cyber-Physical Systems, Artificial Intelligence, Anomaly Detection, Diagnosis, System Reconfiguration

Digital Object Identifier 10.4230/OASIcs.DX.2026.7

Supplementary Material *Dataset*: `https://github.com/imb-hsu/twinflow`

37th International Conference on Principles of Diagnosis and Resilient Systems (DX 2026).
Editors: Ingo Pill, Marina Zanella, and Gregory Provan; Article No. 7; pp. 7:1–7:14

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The realization of robust, resilient and efficient production systems is a central objective in the field of cyber-physical production systems (CPPSs). In this context, various methods for analyzing system operation have been investigated, including anomaly detection, which aims to identify deviations from expected system behavior [9], and fault diagnosis that targets to isolate the root causes of the deviations in order to explain the problem [13, 4]. Although significant progress has been made on these tasks and many methods have been proposed, their evaluation is a quite complex problem [21]. Corresponding experiments are still typically conducted using isolated use cases, task-specific datasets, or simplified system models [12]. This limits the comparability of methods and makes it difficult to assess their suitability for realistic production settings.

One key challenge is the lack of comprehensive labeled evaluation data that enables a realistic, meaningful, and fair assessment [6]. Furthermore, production systems are highly diverse and consist of various interacting components whose integration gives rise to emergent system behavior. As with symbolic approaches based on first-principles reasoning [28, 24] and abductive reasoning [25], this requires a holistic analysis of system operation rather than isolated analysis of individual components [18]. Moreover, new technologies, machines, and devices, such as autonomous mobile robots (AMRs) and advanced industrial robots, are increasingly being introduced into production systems. These developments change the dependencies between system components and variables compared with earlier production systems [8]. As a result, continuously updated benchmarks are needed to evaluate whether proposed solutions remain applicable under evolving conditions [14].

To address this gap, we present *HSU TwinFlow*, a living digital-twin-based benchmark built around a hypothetical production system with *several hundred* components and *several thousand* signals. It encompasses conveyors, industrial robots, autonomous mobile robots (AMRs), and operating stations (see Figure 1). The digital twin captures both the structural layout and the operational logic of a production facility and generates production data from high-level input scenarios that define, for example, product characteristics and component parameters. In addition to nominal operation, the model supports the definition of failure cases affecting different asset types, process stages, and system locations. This allowed us to generate a benchmark dataset comprising representative, labeled scenarios for evaluating anomaly detection and fault diagnosis solutions. This living benchmark can be continuously extended and updated as the digital twin, scenarios, and evaluation requirements evolve.

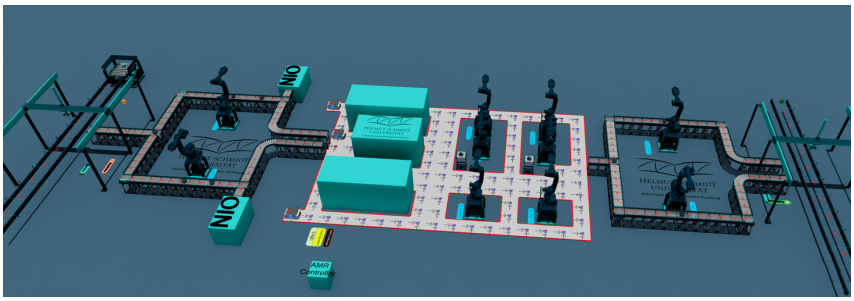

Figure 1 The *HSU TwinFlow* digital twin is implemented in *Emulate3D* and combines physics-based simulation with components from *Digital Twin Factory* and scenario generation by *Novatio*.

The contributions of this work are threefold:

- We introduce a digital twin of a complex production system comprising multiple interacting asset classes, including conveyors, robots, AMRs, and operating stations.
- We provide training and evaluation datasets with both nominal and faulty behavior, following common anomaly detection and diagnosis requirements.
- We establish an initial benchmark by evaluating two representative anomaly detection methods and two representative fault diagnosis methods.

The remainder of this paper is structured as follows. Section 2 presents related work. Section 3 introduces the digital twin, including the considered asset classes and the operational modeling approach. Section 4 defines the benchmark requirements, as well as the evaluation protocols, for anomaly detection and diagnosis. Section 5 presents the benchmark design, including scenario-based data generation and fault modeling, while Section 6 reports the initial experiments and results obtained with selected methods. Finally, Section 7 summarizes the main conclusions, and outlines limitations and future work.

2 Related Work

Benchmark-based evaluation is essential for comparing fault (anomaly) detection and diagnosis methods under reproducible conditions. Established industrial benchmarks, such as the Tennessee Eastman Process, provide controlled process data and have been widely used to evaluate fault detection methods [5, 10]. Other benchmark datasets focus on specific data characteristics or sensing modalities. For example, multivariate time-series benchmarks support the systematic comparison of anomaly detection and diagnosis methods in cyber-physical systems [9], while datasets such as MIMII provide acoustic observations of normal and anomalous behavior in industrial machines [27]. These benchmarks have contributed significantly to method development, but they typically address specific processes, data modalities, or isolated tasks.

The diagnosis community has also established shared evaluation settings through benchmark competitions. The First International Diagnosis Competition (DXC'09) introduced a common framework for comparing diagnosis algorithms on shared systems, scenarios, ground truth, and evaluation criteria [16]. Subsequent editions, including DXC'10, DXC'11, and DXC'13, continued this effort across industrial, synthetic, software, and application-specific tracks [15, 26, 31]. More recently, the DX Competition 2025 revived this benchmark-oriented competition format with several dedicated fault-diagnosis benchmarks [23]. Domain-specific diagnosis benchmarks have also been proposed, for example for water distribution networks [32]. These works demonstrate the value of shared evaluation environments, while also highlighting the need for benchmarks tailored to the structure and dynamics of specific application domains.

In the CPPS domain, existing benchmarks and datasets address important aspects of fault-related tasks in production systems. Some provide simulation environments and datasets for evaluating AI-based methods for fault handling, diagnosis, reconfiguration, and planning [1, 6, 17]. These works are particularly relevant because they consider production-system characteristics such as interacting assets, modular system structures, and operational dependencies. Related work further addresses artificial sensor-data generation for comparing unsupervised learning methods [33], learning physical concepts in CPSs [30], and scalable modeling of time-dependent production-line sensor data, for example using the Bosch Production Line Performance dataset [14]. However, existing benchmarks typically rely on specific system abstractions, simplified scenarios, or assumptions that limit their representativeness for realistic CPPSs.

Digital twins provide a promising basis for addressing this limitation, as they can closely represent real systems while enabling controllable simulation of complex system behavior [19, 3]. They are particularly relevant for production-oriented benchmarks because they can connect operational data with asset structures, process dependencies, and system context. However, reusable benchmark datasets that combine realistic production behavior, structured digital-twin information, and controlled fault scenarios remain limited.

3 *HSU TwinFlow*: A CPPS Digital Twin

The digital twin was developed by combining modules inspired by several existing real-world systems into a representative industrial production system. Its layout is shown in Figure 2. The system consists of five functional areas, each containing specific asset types:

- **Goods input (WE):** The goods input area represents the start of the plant, where raw items enter the system on pallets. It includes two segments of connected chain conveyors, `CC_1-6` and `CC_16-20`, with creation of pallets with input items at one end and a sink for empty pallets at the other end.
- **Singulation area (VZ):** The singulation area separates incoming lots of palletized items into individual items using the portal robot `PR_7`. It also comprises roller conveyors `RC` and chain-lift conveyors `CLT`, which redirect items from one conveyor line to another. Two six-axis robots, `RB_11` and `RB_12`, unpack items at conveyors `RC_34` and `RC_53`, respectively. The implemented logic further separates defective items and directs them to the non-conforming outputs `NIO_1` and `NIO_2`.
- **Autonomous mobile robots area (AMR):** The autonomous mobile robots area follows a matrix-production concept. It consists of three autonomous robots `AMR`, which transport items to any of the six robot stations `RS`. Each robot station has a corresponding robot `RB` that processes items according to instructions defined for each item type. For example, an item may be routed through `RS_1`, then `RS_3`, and finally `RS_2`.
- **Parallel processing area (PAR):** The parallel processing area distributes items across two parallel processing paths. It includes roller conveyors `RC`, chain-lift-transfer conveyors `CLT`, and two robots, `RB_21` and `RB_22`, which perform final processing at `RC_102` and `RC_131`, respectively.
- **Goods output (WA):** The goods output area represents the final output of the plant. Empty pallets arrive on chain conveyors, and a robot places finished products onto them. The loaded pallets then continue through the output area and leave the plant.

The behavior of the complete CPPS is defined by a hybrid simulation model that combines discrete-event production logic with physics-based component behavior. Discrete events represent production and material-flow operations, such as product routing, conveyor transport, robot actions, and fault injection, while continuous component behavior is represented by underlying physics-based simulation models governing variables such as current, speed, and force. As a result, we obtain a rich system-level simulation model of the CPPS.

4 Benchmark Requirements and Protocols

This section first introduces the general process knowledge, followed by the task-specific requirements for anomaly detection and diagnosis and the corresponding evaluation protocols used in the benchmark. The implemented production logic can be represented as a directed material flow graph, in which nodes denote system assets and edges represent possible material transitions. This structural information is made available to the evaluated methods as part of the expert knowledge provided by the benchmark.

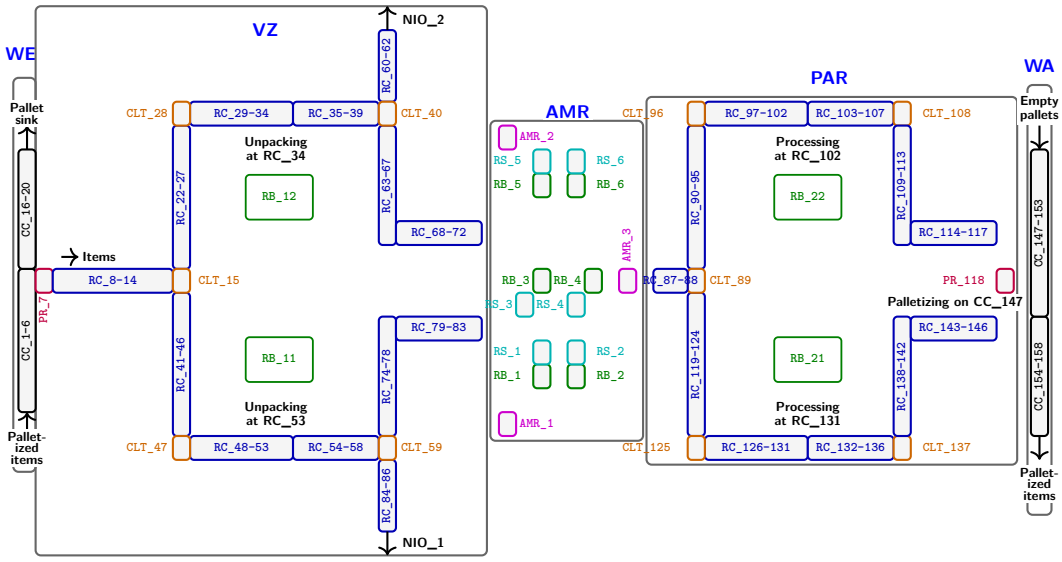

Figure 2 Two-dimensional layout of the *HSU TwinFlow* production system.

The corresponding knowledge graph is provided as a `json` file defining the nodes and edges of the material flow graph. It can be used to compare observed system behavior with the intended system design and to infer possible future material movements based on the current state and position of items in the system. A tiny subset of approximately 1000 variables is plotted in Figure 4.

4.1 Anomaly Detection Requirements and Protocols

The first task targeted by our benchmark is anomaly detection (AD). AD aims to distinguish nominal system behavior from abnormal operating conditions and is commonly categorized into supervised, semi-supervised, and unsupervised settings [2]. For CPPSs, semi-supervised and unsupervised settings are particularly relevant, as CPPSs typically provide data dominated by normal operation [20]. In this work, normal-only training data corresponds to semi-supervised AD, whereas training data that may contain anomalies corresponds to unsupervised AD.[1]

Anomaly detection methods typically compute a continuous anomaly score $as(\mathbf{x}_i)$ for each system observation $\mathbf{x}_i$ at timestamp t_i. A threshold A_{th} is then applied to classify the observation as normal, if $as(\mathbf{x}_i) \leq A_{th}$, or anomalous, if $as(\mathbf{x}_i) > A_{th}$. If a small labeled subset is available, it may support a more informed threshold selection. During evaluation, anomaly detection models are assessed using a labeled test set.

Therefore, we define the anomaly detection requirements for the benchmark as follows:

Req. AD1 The training data shall consist of one or more observation sequences with corresponding timestamps representing nominal system operation. The dataset may contain a small fraction of abnormal data points at unknown timestamps ($\mapsto$ unsupervised AD).

[1] The *HSU TwinFlow* supports both unsupervised and semi-supervised setting based on the selected training data, as will be presented in Section 5.

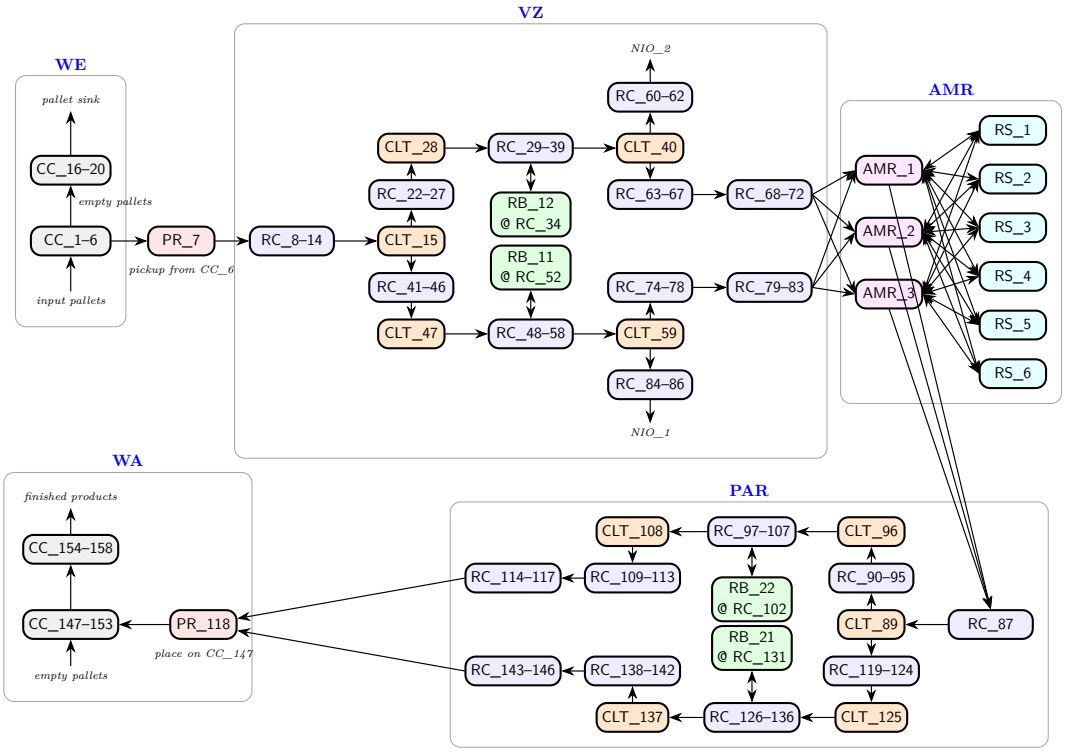

Figure 3 Material flow diagram of the *HSU TwinFlow* production system, showing the main production areas, system assets (components), and directed material flow between them.

Req. AD2 The test data shall provide observation sequences with point-wise anomaly labels a_j for each timestamp t_j.

Req. AD3 *Optional:* A (small) labeled dataset may be provided to support training or threshold selection.

Furthermore, suitable metrics are required to evaluate different anomaly detection approaches. For this benchmark, we use:

1. *Point-wise anomaly detection balanced accuracy*, defined as:

$$BA = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \tag{1}$$

where TP, TN, FP, and FN denote point-wise (for each timestamp) true positives, true negatives, false positives, and false negatives, respectively.

2. *Point-wise anomaly detection F-score*, defined as:

$$F_1 = \frac{2 \cdot P \cdot R}{P+R}, \qquad P = \frac{TP}{TP+FP}, \qquad R = \frac{TP}{TP+FN} \tag{2}$$

where P and R denote point-wise precision and recall, respectively.

3. *F1 composite score* common in time-series anomaly detection [9], defined as:

$$F_{c1} = \frac{2 \cdot P \cdot R_{\text{event}}}{P + R_{\text{event}}} \tag{3}$$

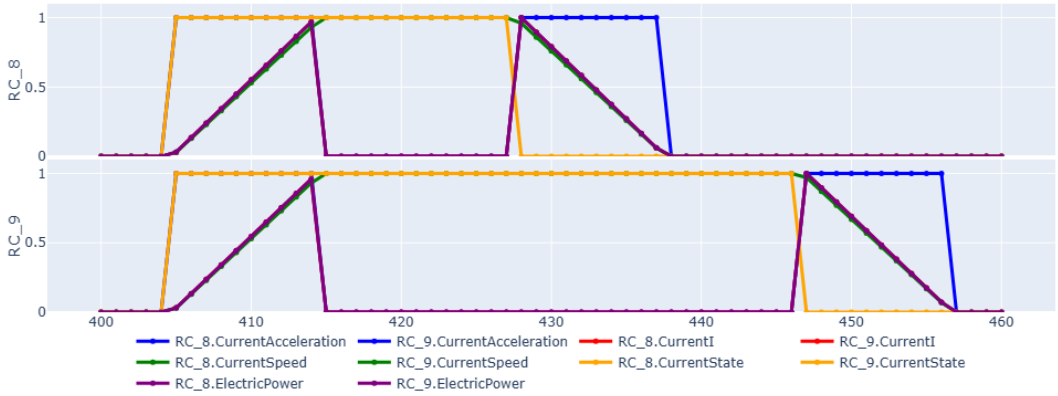

Figure 4 Data of conveyors RC_8 and RC_9 during 1 minute.

where P is point-wise, defined above, and:

$$R_{\text{event}} = \frac{TP_{\text{event}}}{TP_{\text{event}} + FN_{\text{event}}} \tag{4}$$

denotes event-wise recall. Here, TP_{event} is the number of true anomalous segments during which at least one time point is correctly detected as anomalous, while FN_{event} is the number of true anomalous segments missed entirely.

4.2 Diagnosis Requirements and Protocols

Fault diagnosis (DX) aims to explain identified abnormal behavior by determining the affected component(s) and, where possible, the corresponding fault type(s). In the following, let $c \in \mathcal{C}$ denote a system component and let $f \in \mathcal{F}$ denote a fault type. A diagnosis label set $\mathcal{D}$ is represented as a set of tuples $d = (c, f)$, where each tuple specifies that component c is affected by fault type f. Here $\mathcal{D}$ contains at most one tuple for one system component to ensure the mutual exclusiveness of faults:

$$\mathcal{D} \subseteq \mathcal{C} \times \mathcal{F}, \qquad (c, f_1) \in \mathcal{D} \wedge (c, f_2) \in \mathcal{D} \Rightarrow f_1 = f_2. \tag{5}$$

Components that do not appear in any tuple in $\mathcal{D}$ are assumed to be healthy for the considered segment.

Diagnosis approaches commonly use operational data to characterize symptoms, analyze them with respect to expected system behavior or a model, and infer plausible root causes [20]. However, explicit causal or executable simulation models are often not available during diagnosis in the CPPS domain. For this reason, this benchmark supports the approaches that use historical operational data together with structural system knowledge to infer likely diagnosis hypotheses.

We define the diagnosis requirements as follows:

Req. DX1 The training data shall predominantly represent normal system operation to support the analysis of nominal system behavior.

Req. DX2 The test data shall contain both normal and anomalous segments. Each anomalous segment shall be associated with a diagnosis label set identifying the corresponding fault(s).

Req. DX3 *Optional:* Labeled training data containing anomalous segments may be available to support the analysis of system responses to known faults. If provided, each anomalous segment shall be associated with a diagnosis label set.

Req. DX4 A graph shall represent the hierarchical relationships between system components and variables, and shall relate affected components to observed variables.[2]

Req. DX5 For each system component or component type, the available system knowledge shall specify the set of applicable fault types.[2]

In this benchmark, **Req. DX4** is implicitly reflected in the variable identifiers used in the dataset. Each complete variable identifier is formed by combining the component identifier with the corresponding signal name. For example, `RC_48.CurrentSpeed` denotes the `CurrentSpeed` signal of component `RC_48`. Furthermore, **Req. DX5** is addressed by the system knowledge summarized in Table 1.

Table 1 Representative types of faults (failures) for each component type in the *HSU TwinFlow* digital twin.

CC, **RC**, **CLT**, **RS**	**PR**	**AMR**	**RB**
EmergencyStop	EmergencyStop	EmergencyStop	EmergencyStop
MotorFault	MotorFault	MotorFault	MotorFault
SpeedCalibrationFault	SpeedCalibrationFault	SpeedCalibrationFault	SpeedCalibrationFault
WearFault	WearFault	ChargeEmpty	
IncreasedDampingFault	IncreasedDampingFault		
IMeasurementFault			
PeDefect			
PeWrongCalibration			

In this work, diagnosis is evaluated at the segment (sequence) level. Thus, the input to a diagnosis method is a window of operational data, and each segment is associated with a diagnosis label set $\mathcal{D}$. For example, a ground-truth label set may be given as $\mathcal{D} = \{(\texttt{RC\_48}, \textit{MotorFault}), (\texttt{RB\_11}, \textit{SpeedCalibrationFault})\}$, while a method may return the predicted diagnosis set $\hat{\mathcal{D}} = \{(\texttt{RC\_48}, \textit{MotorFault}), (\texttt{RB\_12}, \textit{SpeedCalibrationFault})\}$. For localization performance, only the component identifiers in the tuples are compared, whereas for fault-type performance, also the fault types are considered.

To evaluate diagnosis results, we use the following metrics:

- **Localization balanced accuracy:** Measures whether the method correctly identifies the affected components. For each diagnosis instance, the predicted component set $\hat{\mathcal{C}}$ is compared with the ground-truth component set $\mathcal{C}$, derived from the diagnosis tuple sets $\hat{\mathcal{D}}$ and $\mathcal{D}$.
- **Fault-type balanced accuracy:** Measures whether the method correctly identifies the occurring fault types. For each diagnosis instance, the predicted fault-type set $\hat{\mathcal{F}}$ is compared with the ground-truth fault-type set $\mathcal{F}$, derived from the diagnosis tuple sets $\hat{\mathcal{D}}$ and $\mathcal{D}$.
- **Localization F1-score:** The localization F1-score evaluates the overlap between the predicted and ground-truth affected component sets. It is computed from precision and recall over component labels, treating components in $\hat{\mathcal{C}} \cap \mathcal{C}$ as true positives.
- **Fault-type F1-score:** The fault-type F1-score evaluates the overlap between the predicted and ground-truth fault-type sets. It is computed from precision and recall over fault-type labels, treating fault types in $\hat{\mathcal{F}} \cap \mathcal{F}$ as true positives.

[2] The structural system graph and the mapping from component types to possible fault types are provided by the *HSU TwinFlow* benchmark as separate `json` files.

5 Benchmark Design

The *HSU TwinFlow* follows a scenario-based simulation concept. A scenario specifies a sequence of timed actions and parameter choices that define the execution of a production process. This concept allows the systematic creation of simulation runs under nominal and faulty operating conditions. The scenarios can address individual assets, groups of assets, or complete plant sections.

To address the anomaly detection requirements defined in the previous section, several simulation scenarios are defined. Each scenario specifies when a fault starts, and when the affected component returns to normal operation. The simulations are then executed using these scenario definitions, so that random fault occurrences and repair actions are injected into otherwise nominal system behavior.

In each simulation run, faults (if any) are injected at specified timestamps and repaired after 30 s. Each scenario is defined by a sequence of tuples containing:

- the timestamp at which each fault starts (or ends i.e., when the system is repaired.),
- the affected component,
- the corresponding fault type.

Each file covers a simulated duration of 60 minutes, with operational data sampled with a sampling rate of $f_s = 10\,\text{Hz}$.

Req. AD1 and **Req. DX1** are fulfilled by simulating three training settings, in which 0%, 1%, and 10% of the total simulated time correspond to faulty segments. These settings provide training data ranging from purely nominal operation to data containing a small fraction of randomly placed faulty intervals (see training files in Figure 5).

Req. AD2/Req. DX2 and **Req. AD3/Req. DX3** are addressed by generating faulty scenarios with explicit fault-injection and repair actions (see evaluation files in Figure 5).

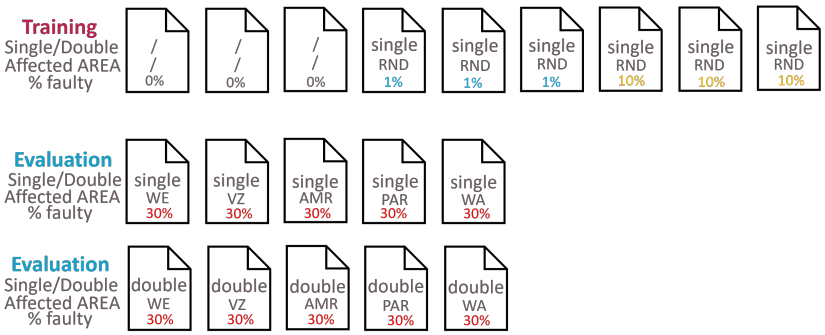

Figure 5 Generated training and evaluation scenarios in the first *Release* of the benchmark datasets.

For evaluation, multiple scenario files are generated. Each file samples faults from one production area only, i.e., from `WE`, `VZ`, `AMR`, `PAR`, or `WA`. Within the selected area, faults are sampled uniformly from the applicable components and fault types. For conveyor-based components, a conveyor line segment (of connected conveyors) is sampled first, instead of directly sampling an individual conveyor. In a second step, one conveyor within the selected line segment is sampled uniformly. This avoids overrepresenting long conveyor lines while still allowing faults to occur on individual conveyor components. The benchmark includes both single-fault and two-fault scenarios. The evaluation scenario sequences are generated such that approximately 30% of each sequence corresponds to faulty behavior.

6 Benchmark Demonstration with Representative Methods

Two anomaly detection methods are considered:

1. Range Monitoring [11]
2. Vanilla Autoencoder [7]

Furthermore, two diagnosis methods are evaluated:

1. Structural-Knowledge-Based Fault Diagnosis [29]
2. Case-Based Fault Diagnosis [22]

6.1 Anomaly Detection

- **Range Monitoring:** Range monitoring is a threshold-based anomaly detection approach. For each observable variable j, an expected operating range is defined by a lower and an upper bound. Given an observation vector $\mathbf{x}_i$ at timestamp t_i, the value of variable j is denoted by $x_{i,j}$. The observation is classified as anomalous if at least one variable falls outside its predefined range:

  $$x_{i,j} < X_j^{\min} \quad \text{or} \quad x_{i,j} > X_j^{\max} \quad \text{for all } j$$

  The method is simple, transparent, and easy to interpret. It is especially suitable for variables with known physical or operational limits. However, range monitoring only detects point-wise deviations and does not capture temporal patterns or correlations between multiple variables.
- **Vanilla Autoencoder:** A vanilla autoencoder is an unsupervised neural-network-based anomaly detection method. It consists of an encoder, which maps an observation vector to a lower-dimensional latent representation, and a decoder, which reconstructs the original input from this representation. During training, the autoencoder learns to reconstruct normal operating data. Given an observation vector $\mathbf{x}_i$ at timestamp t_i, the reconstructed output is given by:

  $$\hat{\mathbf{x}}_i = g_\theta\big(f_\phi(\mathbf{x}_i)\big),$$

  where f_ϕ denotes the encoder, g_θ denotes the decoder, and $\hat{\mathbf{x}}_i$ is the reconstructed observation vector. During evaluation, anomalies are detected using the reconstruction error:

  $$e_i = \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|.$$

  If the reconstruction error e_i exceeds a predefined threshold, the observation at timestamp t_i is classified as anomalous. Compared with range monitoring, the autoencoder can capture nonlinear relationships between variables, but its results are less interpretable.

Table 2 summarizes the benchmark results of the two anomaly detection methods.

Table 2 Benchmark results of the anomaly detection methods.

Method	BA	F1	F1 Comp.
Range Monitoring	Results are available in the public GitHub repository.		
Vanilla Autoencoder	Results are available in the public GitHub repository.		

6.2 Diagnosis

- **Structural-Knowledge-Based Fault Diagnosis:** Structural-Knowledge-Based Fault Diagnosis uses prior knowledge about the organization of the production system to localize faults. The system structure is represented by relationships between production areas, component types, individual components, and observable variables. After an anomaly is detected in one or more signals, the affected variables are mapped to the corresponding component nodes in the structural knowledge model. Based on the identified component type, the set of possible fault types can be retrieved.
  This approach does not require an explicit causal model. Instead, it relies on ownership and hierarchy relations:

  $$\textit{anomalous variable} \rightarrow \textit{component} \rightarrow \textit{component type} \rightarrow \textit{possible faults}$$

  Therefore, this method provides an interpretable baseline for mapping detected anomalies to affected components and candidate fault types.
- **Case-Based Fault Diagnosis:** Case-Based Fault Diagnosis uses previously observed fault cases as reference patterns for diagnosing new abnormal behavior. In this work, the reference cases are extracted from the labeled generated files, which are not used by the first diagnosis method. Each reference case contains an observation vector $\mathbf{x}_r$ (or consecutive vectors) together with the corresponding fault label, affected component, component type, and production area.
  During diagnosis, a test observation vector $\mathbf{x}_i$ is compared with the stored reference cases. The most similar case is retrieved, and its label is used to infer the likely fault type and affected component. A general similarity-based diagnosis step is:

  $$c^* = \arg\min_{c_r \in \mathcal{C}} d\left(\mathbf{x}_i, \mathbf{x}_r\right)$$

  where $\mathbf{x}_i$ is the observation vector, $\mathbf{x}_r$ is the observation vector of a stored case c_r, $\mathcal{C}$ is the set of reference cases, and $d(\cdot,\cdot)$ is a distance or dissimilarity measure. The diagnosis result is derived from the label of the nearest reference case c^*. In this work, $d(\cdot,\cdot)$ is defined as the Euclidean distance.
  Compared with Structural-Knowledge-Based Fault Diagnosis, Case-Based Fault Diagnosis can exploit characteristic multivariate signal patterns. However, its performance depends on the coverage and quality of the available reference cases.

Table 3 summarizes the benchmark results of the diagnosis methods.

Table 3 Benchmark results of the diagnosis methods.

Method	Loc. BA	Fault BA	Loc. F1	Fault F1
Structural-Knowledge-Based	Results are available in the public GitHub repository.			
Case-Based	Results are available in the public GitHub repository.			

6.3 Discussion of Results

The benchmark results provide an initial indication of how the evaluated methods perform on the proposed datasets. The selected anomaly detection and diagnosis methods cover different levels of complexity and interpretability, allowing a first comparison between simple baseline

approaches and data-driven alternatives. Overall, the results illustrate the applicability of the benchmark and provide a basis for more detailed analyses of method behavior, failure cases, and performance differences in future evaluations.

7 Conclusion

The proposed benchmark addresses an important gap between simplified benchmark systems and the requirements of applied and theoretical research on industrial CPPSs. Its main strength lies in combining a representative digital twin model with configurable fault scenarios and scenario-based data generation. This enables the systematic creation of both nominal and faulty operating data under controlled and reproducible conditions.

The benchmark provides the structural knowledge required for evaluating and comparing anomaly detection and diagnosis methods, including a material flow graph, component-level system knowledge, configurable scenarios, and quantitative evaluation protocols. In addition, the benchmark is demonstrated using four representative methods, covering both simple interpretable baselines and more data-driven approaches.

A further strength of the proposed setup is its extensibility. Additional fault types, process variants, operating conditions, and reconfiguration strategies can be incorporated without changing the fundamental benchmark design. This makes the benchmark suitable for future work on robust, adaptive, and explainable methods for real-world CPPSs.

By enabling the controlled generation of normal and faulty operating data, the benchmark provides a practical foundation for evaluating fault-related methods in production systems. Future work will extend the benchmark by integrating additional methods, expanding the set of fault scenarios, system reconfiguration, and further analyzing the influence of variable availability and process variability on detection and diagnosis performance.

Data and Code Availability

To support reproducibility and future evaluations of additional methods, the source code, knowledge graphs, scenario configurations, and documentation of our benchmark are available from a GitHub repository: `https://github.com/imb-hsu/twinflow`. The repository also provides links to the training and test dataset files.

It should be noted that the provided dataset contains measurements of all variables available in the CPPS. This level of observability is not necessarily realistic for a real-world CPPS, where only a subset of variables may be accessible. However, providing the full set of variables allows users to select arbitrary subsets for evaluating anomaly detection or diagnosis methods. For example, this enables experiments that analyze how performance changes as the number of available variables is reduced.

References

1 Kaja Balzereit, Alexander Diedrich, Jonas Ginster, Stefan Windmann, and Oliver Niggemann. An ensemble of benchmarks for the evaluation of ai methods for fault handling in cpps. In *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, pages 1–6. IEEE, 2021. doi:10.1109/INDIN45523.2021.9557516.

2 Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009. doi:10.1145/1541880.1541882.

3 Alexander Diedrich, Christian Kühnert, Georg Maier, Joshua Schraven, and Oliver Niggemann. AITwin: A uniform digital twin interface for artificial intelligence applications. In Daniel Schulz and Christian Bauckhage, editors, *Informed Machine Learning*, pages 41–60. Springer Nature Switzerland, Cham, 2025. doi:10.1007/978-3-031-83097-6_3.

4 Brett Dowdeswell, Roopak Sinha, and Stephen G. MacDonell. Finding faults: A scoping study of fault diagnostics for industrial cyber–physical systems. *Journal of Systems and Software*, 168:110638, 2020. doi:10.1016/j.jss.2020.110638.

5 James J. Downs and Ernest F. Vogel. A plant-wide industrial process control problem. *Computers & Chemical Engineering*, 17(3):245–255, 1993. doi:10.1016/0098-1354(93)80018-I.

6 Jonas Ehrhardt, Malte Ramonat, René Heesch, Kaja Balzereit, Alexander Diedrich, and Oliver Niggemann. An AI benchmark for diagnosis, reconfiguration & planning. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2022. doi:10.1109/ETFA52439.2022.9921546.

7 Benedikt Eiteneuer, Nemanja Hranisavljevic, and Oliver Niggemann. Dimensionality Reduction and Anomaly Detection for CPPS Data using Autoencoder. In *20th IEEE International Conference on Industrial Technology*, volume 2019-Febru, Melbourn, Australia, March 2019. doi:10.1109/ICIT.2019.8755116.

8 Hoda ElMaraghy, Laszlo Monostori, Guenther Schuh, and Waguih ElMaraghy. Evolution and future of manufacturing systems. *CIRP Annals*, 70(2):635–658, 2021. doi:10.1016/j.cirp.2021.05.008.

9 Astha Garg, Wenyu Zhang, Jules Samaran, Ramasamy Savitha, and Chuan-Sheng Foo. An Evaluation of Anomaly Detection and Diagnosis in Multivariate Time Series. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–10, August 2021. doi:10.1109/TNNLS.2021.3105827.

10 Fabian Hartung, Billy Joe Franks, Tobias Michels, Dennis Wagner, Philipp Liznerski, Steffen Reithermann, Sophie Fellenz, Fabian Jirasek, Maja Rudolph, Daniel Neider, Heike Leitte, Chen Song, Benjamin Kloepper, Stephan Mandt, Michael Bortz, Jakob Burger, Hans Hasse, and Marius Kloft. Deep anomaly detection on tennessee eastman process data. *Chemie Ingenieur Technik*, 95(9):1415–1425, 2023. doi:10.1002/cite.202200238.

11 Nemanja Hranisavljevic, Alexander Maier, and Oliver Niggemann. Discretization of hybrid CPPS data into timed automaton using restricted Boltzmann machines. *Engineering Applications of Artificial Intelligence*, 95:103826, 2020. doi:10.1016/j.engappai.2020.103826.

12 Nemanja Hranisavljevic, Tom Westermann, Swantje Plambeck, Henrik Sebastian Steude, Gesa Benndorf, and Oliver Niggemann. A Model Learning Perspective on the Complexity of Cyber-Physical Systems. In Oliver Niggemann, Jürgen Beyerer, and Alexander Diedrich, editors, *Machine Learning for Cyber-Physical Systems*, 2025. doi:10.24405/20025.

13 Inseok Hwang, Sungwan Kim, Youdan Kim, and Chze Eng Seah. A survey of fault detection, isolation, and reconfiguration methods. *IEEE Transactions on Control Systems Technology*, 18(3):636–653, 2010. doi:10.1109/TCST.2009.2026285.

14 Felix Janzen, Alexander Diedrich, and Oliver Niggemann. Modular framework for scalable analysis and modeling of time-dependent industrial sensor data using the example of the bosch production line performance dataset. In *Proceedings of the 36th CIRP Design Conference*, Tokio, March 2026. Elsevier.

15 Tolga Kurtoglu, Sriram Narasimhan, Scott Poll, David Garcia, Lukas Kuhn, Johan de Kleer, and Alexander Feldman. Second international diagnostic competition (dxc'10), 2010.

16 Tolga Kurtoglu, Sriram Narasimhan, Scott Poll, David Garcia, Lukas Kuhn, Johan de Kleer, Arjan van Gemund, and Alexander Feldman. First international diagnosis competition-dxc'09. In *Proc. International Workshop on Principles of Diagnosis*, volume 9, pages 383–396, 2009.

17 Lukas Moddemann, Jonas Ehrhardt, Alexander Diedrich, and Oliver Niggemann. The HAI-CPPS benchmark: Evaluating AI capabilities across hybrid data spaces. In *2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2025. doi:10.1109/ETFA65518.2025.11205680.

18 Laszlo Monostori, Botond Kádár, Thomas Bauernhansl, Shinsuke Kondoh, Soundar Kumara, Gunther Reinhart, Olaf Sauer, Günther Schuh, Wilfried Sihn, and K Ueda. Cyber-physical systems in manufacturing. *CIRP Annals - Manufacturing Technology*, 65:621–641, 2016. doi:10.1016/j.cirp.2016.06.005.

19 Oliver Niggemann, Alexander Diedrich, Christian Kühnert, Erik Pfannstiel, and Joshua Schraven. A generic DigitalTwin model for artificial intelligence applications. In *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 55–62, 2021. doi:10.1109/ICPS49255.2021.9468243.

20 Oliver Niggemann and Volker Lohweg. On the diagnosis of cyber-physical production systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

21 Ingo Pill and Johan de Kleer. Assessing Diagnosis Algorithms: Of Sampling, Baselines, Metrics and Oracles. In Marcos Quinones-Grueiro, Gautam Biswas, and Ingo Pill, editors, *36th International Conference on Principles of Diagnosis and Resilient Systems (DX 2025)*, volume 136 of *Open Access Series in Informatics (OASIcs)*, pages 5:1–5:19, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.DX.2025.5.

22 Ingo Pill, Johan de Kleer, and Franz Wotawa. Challenges for model-based diagnosis. In *Proceedings of the 35th International Workshop on Principles of Diagnosis*, OASIcs, 2024. doi:10.4230/OASIcs.DX.2024.6.

23 Ingo Pill, Daniel Jung, et al. The dx competition 2025 and its benchmarks. In *International Conference on Principles of Diagnosis and Resilient Systems*, 2025.

24 Ingo Pill and Thomas Quaritsch. Rc-tree: A variant avoiding all the redundancy in reiter's minimal hitting set algorithm. In *Proceedings of the 2015 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, ISSREW '15, pages 78–84, USA, 2015. IEEE Computer Society. doi:10.1109/ISSREW.2015.7392050.

25 Ingo Pill and Franz Wotawa. *Fault Detection and Localization Using Modelica and Abductive Reasoning*, pages 45–72. Springer International Publishing, Cham, 2018. doi:10.1007/978-3-319-74962-4_3.

26 Scott Poll, Johan de Kleer, Rui Abreu, Matthew Daigle, Alexander Feldman, David Garcia, and Andy Sweet. Third international diagnostic competition – dxc'11. In *Proc. International Workshop on Principles of Diagnosis*, pages 267–278, 2011.

27 Harsh Purohit, Ryo Tanabe, Kenji Ichige, Takashi Endo, Yuki Nikaido, Kaori Suefusa, and Yohei Kawaguchi. MIMII dataset: Sound dataset for malfunctioning industrial machine investigation and inspection. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop*, pages 209–213, 2019.

28 Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987. doi:10.1016/0004-3702(87)90062-2.

29 H. S. Steude, A. Diedrich, I. Pill, L. Moddemann, D. Vranješ, and O. Niggemann. Data-driven diagnosis for large cyber-physical systems with minimal prior information. In *Proceedings of the IEEE Conference on Artificial Intelligence (CAI)*. IEEE, May 2026.

30 Henrik Steude, Alexander Windmann, and Oliver Niggemann. Learning physical concepts in cps: A case study with a three-tank system. *IFAC-PapersOnLine*, 55(6):15–22, 2022.

31 Andy Sweet, Alexander Feldman, Sriram Narasimhan, Matthew Daigle, and Scott Poll. Fourth international diagnostic competition – dxc'13. In *Proc. International Workshop on Principles of Diagnosis*, pages 224–229, 2013.

32 Anna Sztyber, Elodie Chanthery, and Louise Travé-Massuyès. Benchmark for fault diagnosis of water distribution network. In *The 34th International Workshop on Principles of Diagnosis (DX'23)*, 2023.

33 Bernd Zimmering, Oliver Niggemann, Constanze Hasterok, Erik Pfannstiel, Dario Ramming, and Julius Pfrommer. Generating Artificial Sensor Data for the Comparison of Unsupervised Machine Learning Methods. *Sensors*, 21:2397, 2021. doi:10.3390/s21072397.