# Automatic Model Generation and Data Assimilation Framework for Cyber-Physical Production Systems

Wen Jun Tan
Moon Gi Seok
Wentong Cai
wjtan@ntu.edu.sg
School of Computer Science and Engineering
Nanyang Technological University
Singapore

## ABSTRACT

The recent development of new technologies within the Industry 4.0 revolution drives the increased digitization of manufacturing plants. To effectively utilize the digital twins, it is essential to guarantee a correct alignment between the physical system and the associated simulation model along the whole system life cycle. Data assimilation is frequently used to incorporate observation data into a running model to produce improved estimates of state variables of interest. However, it assumes a closed system and cannot handle structural changes in the system, e.g., machine breakdown. Instead of combining the observation data into an existing model, we aim to automatically generate the model concurrently with the data assimilation procedure. This can reduce the time and cost of building the model. In addition, it can generate a more accurate model when sudden operational changes are not reflected at the higher planning levels. Component-based model generation approach is used with the application of data and process mining techniques to generate a complete process model from the data. A new data assimilation method is proposed to iteratively generate new models based on the arrival of further data. Each model is simulated to obtain the system performance, which will be compared to the real system performance to select the best-estimated model. Identical twin experiments of a wafer-fab simulation are conducted under different scenarios to evaluate the feasibility of the proposed approach.

## CCS CONCEPTS

• **Computing methodologies → Data assimilation**; *Modeling methodologies*; • **Software and its engineering → Petri nets**; • **Applied computing → Industry and manufacturing**.

## KEYWORDS

Automatic Model Generation, Data Assimilation, Cyber-Physical Production Systems

*SIGSIM-PADS '23, June 21–23, 2023, Orlando, FL, USA*
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0030-9/23/06.
https://doi.org/10.1145/3573900.3591112

**ACM Reference Format:**
Wen Jun Tan, Moon Gi Seok, and Wentong Cai. 2023. Automatic Model Generation and Data Assimilation Framework for Cyber-Physical Production Systems. In *ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '23), June 21–23, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3573900.3591112

## 1 INTRODUCTION

The recent developments in Industry 4.0 drive increased research attention on Cyber-Physical Production Systems (CPPS) due to increasing production intricacy in a more demanding market. CPPS are the collaborations of the physical manufacturing systems and their ongoing processes with their virtual counterparts on the computing systems to provide data-accessing or data-processing services [11]. To effectively utilize these digital twins for optimization, predictive and prescriptive analyses, it is essential to guarantee a correct alignment between the physical system and the associated simulation model along the entire system life cycle [18].

Data assimilation has been used to assimilate observed data into the simulation model to produce a time sequence of estimated system states and iteratively select the best system estimation [3]. The initial concept of data assimilation is mainly used for the initialization of a simulation model of a closed system [30]. However, changes in the observation data may also result from dynamic structural changes in the system. This means that the original simulation model built from the past system will be invalid; initialization of an outdated model will render the simulation results useless. Instead of assimilating the observation data into an existing model, we propose to create the model concurrently with the data assimilation procedure.

However, the creation of these simulation models can be time-consuming and costly [19]. Setting up and executing a simulation model is often a manual process that involves several steps and expertise – e.g., consolidating and preparing input data for the simulation by a data analyst, and implementing simulation models by a simulation expert [4]. Hence, the development cycles of simulation models tend to take several days to weeks [16]. In addition, sudden operational changes due to unforeseen circumstances such as machine breakdown may not be reflected at higher planning levels, e.g, long-term planning. To reduce the modeling efforts and track the changes at the operational levels more closely, a possible strategy is to automate the generation of the simulation models.

A new approach to combine data assimilation with automatic model generation is proposed to handle dynamic structural changes in a smart manufacturing system. An automatic component-based model generation is utilized to generate the simulation model from the data. Each newly observed event is aggregated into an event log. Concurrently, the system performance of the manufacturing system is also observed, e.g., factory throughput. First, the basic components of a manufacturing system are designed to reflect the functionalities of the real system. Data and process mining techniques are applied to the event log to extract the basic components to form a process model. A windowed event log is used to adapt to changes in the system. The generated process model is simulated to obtain the measured performance. Next, data assimilation will be performed by comparing the simulation performance with the real system performance to generate new models, and the best-estimated simulation model will be selected. By iteratively generating new process models and selecting the best-estimated model, the proposed approach is able to adapt to changes in the system structure while generating a calibrated simulation model at the same time.

Our contributions can be summarized as follows:

- **Automatic Model Generation**: Component-based model generation approach is used with the application of data and process mining techniques to generate a process model of a generic manufacturing system.
- **Model Adaption using Data Assimilation**: A new data assimilation method is proposed to handle structural changes in the system by generating new models automatically from the windowed event log and selecting the best-estimated model. The model fitness is evaluated by comparing the historical observations with the simulated measurements within the time window.

The rest of the paper is organized as follows: Section 2 explores the related works and highlighted the differences. Section 3 introduces the problem settings regarding the system model, observation data, and process model that is generated. Section 4 describes the proposed automatic model generation and data assimilation framework. Section 5 validates the automatic model generation and evaluates the data assimilation procedure. Section 6 concludes the paper and discusses future works.

## 2 RELATED WORKS

### 2.1 Automatic Process Generation

Automatic model generation is defined as a method of creating simulation models from external data sources using interfaces of the simulation system and algorithms [13]. These methods can be classified into three main categories [2, 10]:

- **Parametric approaches**: Models are initialized and configured based on parameters,
- **Structural approaches**: Model generation is based on data describing the structure of a system,
- **Hybrid-knowledge-based approaches**: combining existing methods (expert systems, neural network) with two of the above presented.

Our proposed framework applies an alternative approach by extracting the parameters and structure from the data to generate the model.

Son and Wysk [26] presented an automatic simulation model generation using a shop floor resource model and control model. The control model is constructed logically from a connectivity graph of the resource models. A component-based model generation with a data-driven approach is proposed by [10]. Data analysis results are used to extract the structural and behavioral preconditions as the basis for constructing the model and initializing the model state. The model is adjusted in an iterative process where the data from the real system and the simulation output data are compared and analyzed. However, these approaches require data describing processes and the topology of the system. By utilizing process mining techniques, our approach can extract the system structure from the data.

Krenczyk et al. [13] proposed a semi-automatic concept for model generation based on a parameter-based approach. Parameters will be used to generate different variants of production. Mages et al. [19] proposed a proof of concept to automate and hence reduce the cost of the process of simulation model creation, thereby allowing for the creation of a larger number of selectable solution variants. They used component-based synthesis using combinatory logic to synthesize a product line of varying simulation models for a given configuration to be executed and evaluated to find suitable solutions. However, these parameter-based approaches require the definition of the simulation model, which will be used to generate different variations of the production. Other than automatically extracting parameters from the data, our proposed approach is also able to generate the complete process model.

Friederich et al. [6] proposed the use of process mining techniques to support the extraction of reliability models from event data generated in smart manufacturing systems. More specifically, a stochastic Petri-net is extracted to analyze the overall system reliability as well as to test new system configurations. An online data-driven approach is not considered in their work; only an offline approach is used to extract the process model for further predictive analysis.

### 2.2 Online Model Validation

After generating the process model, it is important to validate the generated model against the system. This can also be used to select the best-estimated model during the data assimilation process. We focus on *operational validation*, which is defined as "determining that the model's output behavior has a satisfactory range of accuracy for the model's intended purpose over the domain of the model's intended applicability" [23].

Lugaresi et al. [17] presented a new method to perform online validation based on harmonic analysis which is able to achieve reliable results even when applied to a relatively small dataset. The proposed method uses information about the power spectral density of KPI trends as signals to detect potential deviations between the model and the real system. Morgan and Barton [21] proposed a methodology to discriminate between two systems based on the Fourier transform of the output trajectory. By analyzing the output in the frequency domain, the method can consider the temporal evolution of data, rather than comparing aggregated values such as the average. Lugaresi et al. [18] proposed a new technique to deal with a limited data set and analyze the evolution in time of the behavior. The simulation model validity is assessed by measuring the similarity between real system data and simulation model data using a sequence comparison technique. We proposed to use coefficient determination of the historical output between the real system and the simulation model to assess the fitness of the model. However, a detailed analysis of our approach compared to the existing methods has not been investigated in this paper. This will be one of the future directions for further investigation.

## 2.3 Data Assimilation

For discrete event simulation, conventional data assimilation methods, such as Kalman filters [28] and Extended Kalman filters [8], are inapplicable due to nonlinearity in the model, and non-Gaussian property of noise. Sequential Monte Carlo (SMC) methods, also called particle filters, can approximate arbitrary probability densities and have little or no assumption about the properties of the system model [1]. Particle filters estimate the posterior distribution of a state trajectory by a set of random samples with associated weights and compute estimates based on these samples and weights. As the number of samples becomes very large, this becomes an equivalent representation to the usual functional description of the posterior distribution [5, 28].

Hu and Wu [9] proposed a data assimilation framework for discrete event simulation. A mechanism is designed to synchronize the system state of a discrete event simulation with the data assimilation time; otherwise, the system state at the data assimilation time will be inaccurate. To address the low diversity of samples due to the deterministic or small degree of stochasticity in the discrete event model, sample rejuvenation is used to add stochastic perturbations to the system states represented by samples after the resampling step.

Xie and Verbraeck [30] presented a particle filter-based data assimilation framework for discrete event simulations. Instead of synchronizing the system state with the data assimilation, they proposed the interpolation of the discrete model states to obtain updated state values. The sequential importance sampling algorithm is applied to update the samples that approximate the posterior distribution of the state trajectory with variable dimensions.

However, the SMC method assumes that model components do not change over time (i.e., closed systems). Our proposed approach sequentially generates new models with the arrival of new data, hence it is not possible to directly apply the existing methods in our framework. The proposed approach can also deal with the diversity of samples. Variations in the samples are introduced when each sample is generated based on a different snapshot of the events and the initial state of the generated model is also randomly generated.

## 3 PROBLEM SETTINGS

## 3.1 System Model

A general production system is modeled with a queue before each resource, as shown in Figure 1. Resources represent the Automated Guided Vehicles (AGVs), assembly cells, or various machines and tools in the factory. Each order will be processed by a sequence of resources in the system according to the process recipe. When the order arrives at the resource, it enters the queue waiting to be processed, emitting the `Arrive` event. If the resource is idle, the order in the queue will be processed, emitting the `Produce` event. Depending on the capacity of the resource, it is possible to process one or more orders concurrently. When the order is processed by the resource, the `Route` event is emitted to indicate that the order is completed and will be routed to the following resource for processing. Reentrant manufacturing is also supported where the orders can repeatedly pass through the same resource at different stages of the process routes.

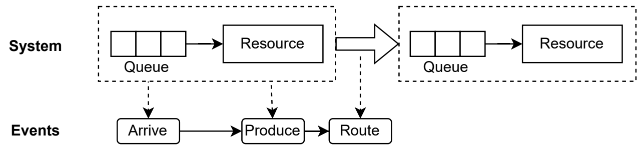


**Figure 1: A general production system and events emitted.**

## 3.2 Observation Data

The manufacturing system's data is collected and distributed to the company's information systems. Several information systems in the company may collect information about the physical system, such as Supervisory Control and Data Acquisition (SCADA), Programmable Logic Controller (PLC), Manufacturing Execution System (MES), or Enterprise Resource Planning (ERP) [6]. The data captured by such systems is then aggregated in event logs or used to generate key performance indicators (KPIs). A manufacturing KPI is a well-defined measurement to monitor, analyze and optimize manufacturing processes regarding their quantity, quality as well as different cost aspects. They give manufacturers valuable business insights to meet their organizational goals.

The general assumptions [15] about a process are slightly modified to fit the manufacturing domain:

- A manufacturing process is triggered by production *orders*.
- An *order* is characterized by a sequence of activities that begin and end by *events*.
- Each event has a corresponding *timestamp* and *resource* that is utilized and released.

Considering the above-mentioned assumptions, an event log $EL$ is defined as a set of event entries:

$$EL = \{e_i | i = 0, 1, ...\} \tag{1}$$

Each event log entry is defined as a tuple $e_i = (t, o, r, c)$, where $t$ is the timestamp, $o$ is the set of order identifiers, $r$ is the resource identifier, and $c$ is the event class. Order identifiers are unique strings representing individual production orders. Event classes are unique strings to mark the basic operations in a production system, i.e., $c \in \{Arrive, Produce, Route\}$.

Typical event logs, used by mainstream process mining techniques, require an event to be related to a *case* or an *object*. However, batch-processing machines can process several orders as a batch at the same time. To aggregate data regarding batch processing in the event logs, *object-centric event logs* (OCEL) are used to relax the assumption that an event is related to exactly one case [7]. Hence, $o = \{o_j | j = 0, 1, ...\}$ so that an event can be related to multiple orders to represent the batches.

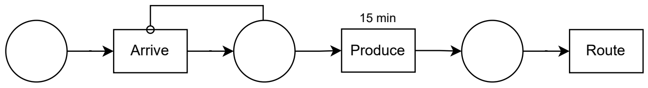


**Figure 2: Petri-net model of activities in a single resource.**

## 3.3 Component-based Process Model

A *Petri-net* model is used to describe production process models extracted from the event log. It is a class of discrete event dynamic systems, which can be used to simulate discrete event systems, such as the production process. Formally, a Petri-net is defined as a tuple $N = (P, T, A)$ where

(1) $P$ and $T$ are disjoint finite set of *places* (drawn as circles) and *transitions* (drawn as rectangles), respectively.
(2) $A \subseteq (P \times T) \bigcup (T \times P)$ is a set of directed arcs.

Each transition is defined as $T_i = (c, r)$, which corresponds to events $(t, o, r, c)$ in the event log. The transition types are modeled after the components in the system. Transitions are labeled as a tuple $(r, c)$ for the resource $r$ and the event class $c$. Figure 2 shows a Petri-net model of a single resource that contains the following components as the transitions: `Arrive`, `Produce`, and `Route`. The `Arrive` transition represents the arrival of a token at an unbounded resource queue in the production process. An inhibitor arc is used to model the resource capacity by preventing a transition from firing based on the number of tokens in the place. The timed transition at the `Produce` represents the production time of the resource. `Route` is represented by an immediate transition that routes the token to the next resource. By simulating the Petri-net model as a discrete event simulation, it is possible to measure various system performances, e.g., KPIs such as the cycle time or throughput of the manufacturing system.

## 4 AUTOMATIC MODEL GENERATION AND DATA ASSIMILATION FRAMEWORK

In this section, the proposed automatic model generation and data assimilation framework is presented. There are two main parts in the framework: (1) automatic model generation for production systems, and (2) data assimilation procedure for automatic model adaption. The first part focuses on generating the component-based process model from the event logs. The second part utilizes data assimilation to incorporate observations from the real system to select the best-estimated process models that are generated.

Figure 3 shows the overview procedure for the data assimilation framework. Observations refer to the data that is observed from the real system. Measurements refer to the data that is measured from the generated model. Two types of information are captured from the *real system*: events and observations. The events are aggregated into the event log and used by the *data and process mining* to extract the corresponding *process model*. The process model is then simulated to obtain the measurements, which are compared to the observations to evaluate the fitness of the generated model. This is carried out in an iterative approach to continuously observe and generate models in order to handle structural changes in the system.

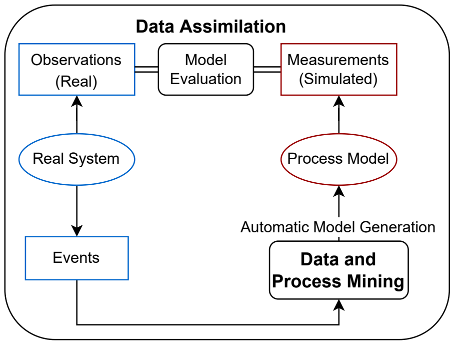


**Figure 3: Automatic model generation and data assimilation framework.**

## 4.1 Automatic Process Model Generation

A set of data and process mining techniques are used to automatically generate the process model based on the system components. First, *process discovery* is used to generate a process model that generalizes the observed behavior from the event data without any apriori information [27]. The discovered process model may describe only the control flow of events or other operational aspects, e.g., processing time. Next, *process enhancement* using various data mining techniques is performed to improve the current process model to achieve a more complete model behavior from the events, e.g., queuing or batching operations [24]. Figure 4 shows the main steps in the data and process mining.

(1) The incomplete processes are trimmed from the OCEL.
(2) Object-centric directly-follows multigraphs (OCDFG) are generated by discovering Directly-Follows Graphs (DFG) from the OCEL.
(3) Process discovery generates the *Petri-net* model from the flattened EL.
(4) *Production time distributions* are fitted based on the activity durations from the EL.
(5) *Resource capacities* are extracted from the OCDFG.
(6) *Decision flows* in the Petri-net are constructed using decision tree classifiers based on the attributes of the events in the event log.

The steps will be explained in further detail in the subsequent subsections.

*4.1.1 Removal of Incomplete Processes.* For an online data assimilation procedure, the sampling of the events may occur for any sequence of events. In Figure 5, there are four orders in which each order is processed in sequence. Hence, the complete process for each order will be the following events: $e_1 \rightarrow e_2 \rightarrow e_3$. Orders $o_1$ and $o_2$ are only able to capture the incomplete processes, while orders $o_3$ and $o_4$ managed to capture the complete processes. Process mining on the partial processes will generate an incorrect model as the process does not begin on events $e_2$ or $e_3$. To remove the incomplete process events, the frequencies of the occurrences of the first events grouped by the resource and event class are determined from the event logs. As the event log aggregate more complete processes, the frequency of complete processes will be higher than

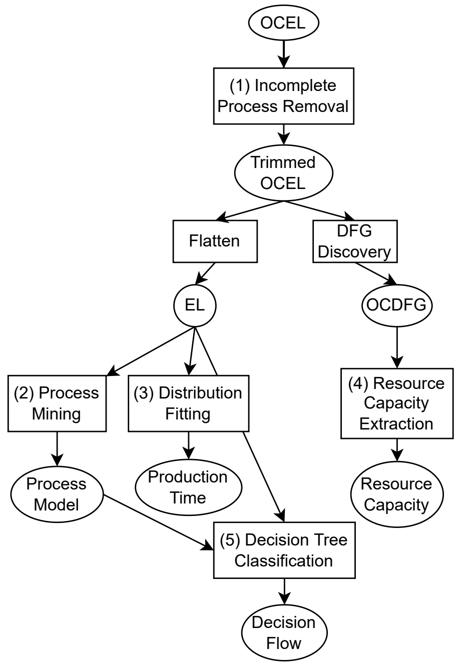


**Figure 4: Steps in the automatic model generation – the ellipse represents the models discovered from the data and the rectangle represents the method to extract the models.**

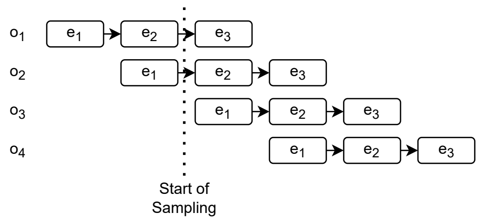


**Figure 5: Sequence of events for each order.**

the incomplete processes, i.e., highly frequent first events indicate complete processes while infrequent first events are incomplete processes. Hence, by filtering infrequent first events from the set of first events, incomplete processes can be removed from the event log before extracting the process model based on the event log. For example, the frequency count of events $\{e_1, e_2, e_3\}$ is $\{2, 1, 1\}$ in Figure 5. If we filter away infrequent events $e_2$ and $e_3$, event $e_1$ is left, which represents the first event of a complete process.

*4.1.2 Process Model Discovery.* This phase is mainly focusing on extracting the whole process structure representing the flow of the order through the manufacturing system. First, the OCEL is flattened to a traditional event log by duplicating the events such that each event relates to only one order. OCEL is mainly utilized in the extraction of resource capacity. Based on the flattened event log, we apply the *Heuristics Miner* [29] to extract the Petri-net model that represents the material-flow process of the manufacturing system. Heuristics Miner can handle noise in the data and find common constructs in the process model.

After generating the basic Petri-net model, the components are identified in the process model by analyzing the structural relationship between the activities. First, the performance (i.e., activity durations) is mined from the event logs by using a replay technique on the Petri-net model and then assigning performance to the arcs. There will be timing information for the arcs `Arrive` → `Produce` and `Produce` → `Route` due to the waiting time in the queue and the production time respectively. Next, the immediate transitions (i.e., transitions with no timing information) are identified to represent the routing decisions. The timed transition before the `Route` represents the operation activity in the resource. Finally, the `Arrive` transition before the `Produce` represents the queuing activity, which will make use of inhibitor arcs to represent resource capacities.

*4.1.3 Production Time Distribution.* The production time for the `Produce` transition is modeled using probability distributions. The activity durations are extracted using the replay technique from the previous phase. Commonly-used distributions in manufacturing for describing activity durations, *Normal* and *lognormal*, are fitted to the extracted activity durations for the transition. The sum of squared errors (SSE) is used to evaluate the goodness of fit between the data and the fitted distributions. The probability distribution with the lowest SSE is selected to be the probability distribution function of the `Produce` transition.

*4.1.4 Resource Capacity Extraction.* Since each resource has the capacity to process one or more orders, it is necessary to extract the processing capacity for the resources for accurate behavior modeling. DFG are graphs where the nodes represent the events/activities in the log and directed edges are present between nodes if there is at least a trace in the log where the source event/activity is followed by the target event/activity [14]. A DFG for each order type can be constructed from the flattened event log. However, we are interested in the capacity that a resource can process for all the order types. Hence, an *object-centric directly-follows multigraph* (OCDFG) is constructed from the OCEL, which is a composition of DFGs of all order types. The number of orders processed by each activity is also recorded in the OCDFG. The maximum and minimum number of orders processed by each activity is extracted to obtain the processing capacity of each resource.

*4.1.5 Decision Flow Extraction.* To model the reentrant production system, the decision flow between the resources needs to be extracted from the event log. After discovering the Petri-net model, the places with branches are identified in order to determine the decision flow. These places are identified as the *decision points*. The features necessary to determine the decision flow are the *order type* and *production step*. The order type can be retrieved for each order by cross-referencing the order identifier from the ERP. Each order type is assumed to utilize a different recipe, i.e., each order type has its production sequence that the order will be processed. The production step is determined by counting the number of *Producing* activities that have been performed on the order. As the order can be reprocessed at the same machine (i.e., in a reentrant production

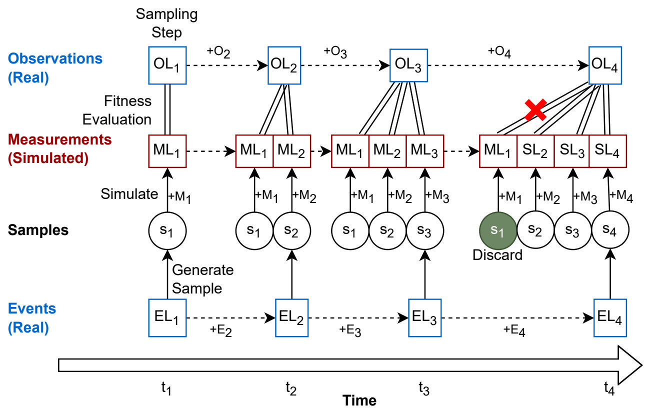


**Figure 6: Overview of data assimilation procedure with a limit of 3 samples.**

**Algorithm 1** Pseudo-code of a sampling step at time $t$

```
Require: Events $E_t$ and Observations $O_t$
1:  $OL_t \leftarrow OL_{t-\Delta T} \bigcup O_t$
2:  $EL_t \leftarrow EL_{t-\Delta T} \bigcup E_t$
3:  if $E_t = \emptyset$ or $O_t = \emptyset$ then
4:      return
5:  end if
6:  $\mathbf{m} \leftarrow \text{GenerateModel}(EL_t[t-w:t])$
7:  $\sigma_{t-\Delta T}, ML_{t-\Delta T} \leftarrow \text{Simulate}(\mathbf{m}, \sigma_0, t-\Delta T)$
8:  $s \leftarrow (\mathbf{m}, \sigma_{t-\Delta T}, ML_{t-\Delta T})$
9:  $\mathcal{S}_t \leftarrow \mathcal{S}_{t-\Delta T} \bigcup s$
10: for all $s \in \mathcal{S}_t$ do
11:     $\sigma_t^s, M_t^s \leftarrow \text{Simulate}(\mathbf{m}^s, \sigma_{t-\Delta T}^s, t)$
12:     $ML_t^s \leftarrow ML_{t-\Delta T}^s \bigcup M_t^s$
13:     $R_s^2 \leftarrow det(OL_t[t-w:t], ML_t^s[t-w:t])$
14: end for
15: if $|\mathcal{S}_t| > N_s$ then
16:     $\mathcal{S}_t.\text{sortBy}(R^2)$
17:     $\mathcal{S}_t.\text{removeLowest}(R^2)$
18: end if
```

system), the *production step* is necessary to determine the routing of the order to the next resource.

The training samples are constructed based on the attributes extracted from the events associated with each edge from the decision point to the next transition in the Petri-net model. The class labels are the subsequent transitions from the decision points. A decision tree is fitted with the training data to predict the path the decision point will transit to based on the event attributes. Gini impurity is used to determine how the features of a dataset should split nodes to form the tree. The decision tree is then used by the execution context to determine the execution of the transition. The transition probability is not used as precise decision flow is necessary to model the exact sequence of production according to the recipe.

## 4.2 Data Assimilation for System Adaption

The behavior of discrete-event simulation is highly non-linear, non-Gaussian [31]. SMC methods can be applied to discrete-event simulation since they are able to approximate arbitrary probability densities and have little or no assumption about the properties of the system [1]. However, the sequential sampling process assumes a closed system, and cannot handle structural changes in the system [30].

A new data assimilation method is proposed to handle changes in the system, as shown in Figure 6. A set of samples $\mathcal{S}$ is used to approximate the system, where each sample consists of a generated simulation model, the corresponding simulation states, and the historical measurements of the model. At each sampling step, the events from the real system are used to generate a new sample. The sample is simulated to obtain the measurements. Instead of comparing the likelihood between the observations from the real system and simulated measurements at each sampling step, the coefficient of determination ($R^2$) between the historical observations and measurements is used to evaluate the fitness of the generated models. $R^2$ measures how well the observed outcomes are replicated by the model, based on the proportion of total variation of outcomes exhibited by the model. The proposed method then continuously generates new samples with the arrival of new events and observations, while discarding samples with low fitness.

For example in Figure 6, the event log $EL_1$ is used to generate sample $s_1$, which is simulated until time $t_1$ to obtain measurements $ML_1$. The fitness of $s_1$ is evaluated by comparing the measurements $ML_1$ with the observations $OL_1$. In the next sampling step $t_2$, the new events $E_2$ are appended to $EL_1$ to obtain $EL_2$, which is used to generate a new sample $s_2$. The samples $s_1$ and $s_2$ are simulated until time $t_2$ to obtain the corresponding new measurements, $M_1$ and $M_2$, which are appended to the respective historical measurements $ML_1$ and $ML_2$. At the same time, the new observations, $O_2$, from the real system are appended to $OL_1$ to obtain $OL_2$, which is then used to evaluate the samples $s_1$ and $s_2$ by comparing to the historical measurements $ML_1$ and $ML_2$. The same procedure is repeated for every sampling step. As there is a limit of 3 samples, at time $t_4$, the sample with the lowest fitness, i.e., $s_1$, is discarded.

*4.2.1 Model Generation and Fitness Estimation.* Algorithm 1 shows the pseudo-code of generating samples at sampling time $t$. Each sample $s$ is defined as a tuple $s = (\mathbf{m}, \sigma, ML)$, representing the generated model $\mathbf{m}$, the simulation state $\sigma$, and measurements $ML$. $ML$ represents the list of historical measurements of the generated model. First, the set of events $E_t$ and observations $O_t$ are aggregated from the real system from time $t - \Delta T$ to $t$. $E_t$ and $O_t$ are appended to the previous event log $EL_{t-\Delta T}$ and observation list $OL_{t-\Delta T}$ respectively. In a discrete event simulation, the state values are only updated when events happen, so it is possible the $E_t$ or $O_t$ are empty. In this case, the sample generation for this sampling step can be skipped (line 4).

To generate new models that reflect the changes in the system, a window of events from the event log $EL_t[t - w : t]$, which represents events between time $t - w$ to time $t$, is used to generate model $\mathbf{m}$ (line 6). $w$ defines the period of the time window. The time window is used to limit the number of events used for model

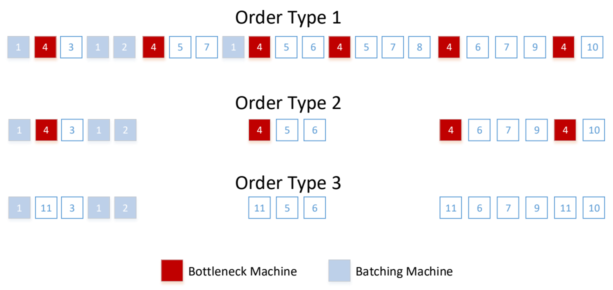


**Figure 7: Process recipe for each order type.**

generation such that older events from an outdated system will not be included in the event log. The model generation is based on the method described in Section 4.1. The newly generated model $\mathbf{m}$ is simulated from a random initial state $\sigma_0$ to the time step $t - \Delta T$, and the corresponding measurements are obtained from the simulation. The generated model $\mathbf{m}$, simulation state $\sigma_{t-\Delta T}$ and measurements $ML_{t-\Delta T}$ forms the new sample (line 8). The new sample is appended to the set of samples $\mathcal{S}_t$.

Each sample in $\mathcal{S}_t$ is measured by simulating it from time step $t - \Delta t$ to $t$ (line 11). The measurements from each sample are aggregated with the previous measurements (line 12). $R^2$ is used to determine the fitness of the sample by measuring how well the historical observations ($OL$) match the measurements of the sample ($ML^s$) (line 13). Similarly, windowed observations and measurements are used to limit the time period for comparison.

To maintain a set of $N_s$ samples, if the number of samples is larger than $N_s$ (line 15), the samples are then sorted according to $R^2$ and the sample with the lowest $R^2$ is removed.

*4.2.2 Terminating Condition.* The distribution of the $R^2$ of the samples can be used to terminate the data assimilation process. An $R^2$ of 1 indicates that the measurements of the generated model perfectly fit the observations. Hence, as the $R^2$ of the samples approaches 1, this means that the current set of samples has similar system output as the observed system. If $R^2$ only converges (yet not approaching 1), this means that the process mining reaches the limits of discovery, and cannot generate better models to capture the behavior of the system.

## 5 EXPERIMENTS

Due to the difficulties of obtaining the data from a real-world factory, a simulation of a wafer fab is used in place of the real system. First, the generated model is validated to ensure the correctness of the simulated output compared to the system output. Next, the identical-twin experiments are used to evaluate the effectiveness of the proposed framework. In the identical-twin experiment, a simulation is first run and the corresponding data is recorded. Lastly, structural changes to the system are emulated to showcase the applicability of the data assimilation framework.

### 5.1 Case Study - Wafer Fabrication Factory

The case study utilizes a scaled-down representation of a real-world wafer-fab at *Harris Semiconductor* described by [12]. This model was developed for a discrete-event simulation, which is an important tool for decision-making in the wafer fab [20]. It is based on an actual wafer-fab factory and captures the main complexities of the actual systems while remaining compact enough to be simulated in a reasonable amount of time.

*Lots* are the moving entities in the wafer fab. Each lot contains a fixed number of wafers. The lots are generated based on customer orders. In the wafer fab, the lots are processed in a series of process steps using multiple pieces of equipment.

A single *wafer fab* is composed of several work areas. These areas are used for wafer processing and sorting in a clean room environment. A single *work area* typically contains dozens of different work centers. The *work centers* of a work area are closely related logically or physically. A single work center is a set of machines that provide similar processing capabilities. Work centers are also called *machine groups* or *tool groups*. Some tools can process only a single wafer at a time, while batch processing tools can process a set of lots simultaneously.

The wafer fab is modeled as a set of event-driven model components whose state changes are triggered by events [25]. The time unit used in the simulation is *minutes*. The typical model components consist of machines and wafer-lot queues. The lot is modeled as a simulation event, and the process flow describes how a lot is routed among the machines. Each simulation event contains information regarding the lot name, the number of wafers, and process recipe, etc.

Each machine model contains input and output ports to receive simulation events and send the events to the next machine model. When the machine model receives a lot, it performs a specific process according to a recipe of the lot that determines the operation parameters such as the setup time, processing time, etc. After processing, the lot is transferred to the next machine in the process flow. Each machine has a wafer-lot queue, which is a buffer to store the incoming lots. The waiting lots in the buffer are dispatched to available machines based on pre-defined dispatching rules, such as the First-in-First-Out (FIFO) dispatching rule. The lots are also assumed to be instantaneously transferred between successive process steps. By coupling the machine models using input and output ports, the modeler can build various types of wafer-fab systems using modular model components.

There are 11 machine groups and 3 order types in this simulation model, as shown in Figure 7. The number of operations for Order Types 1, 2, and 3 is 22, 14, and 14, respectively. The bottleneck station of the fab (`Station 4`) is visited repeatedly, which corresponds to the photolithography process. This bottleneck is mainly due to the costly and scarce resources for the photolithography process.

The wafer-fab simulation is used as the system to generate the events and observations. The wafer lots and machines represent the *orders* and *resources* respectively in the production system. Every time a wafer lot enters a wafer-lot queue, the `Arrive` event is emitted. When the waiting lots are dispatched to the machine for processing, the `Produce` event is emitted. When the lot finishes processing at the machine, a `Route` event is emitted to signify the

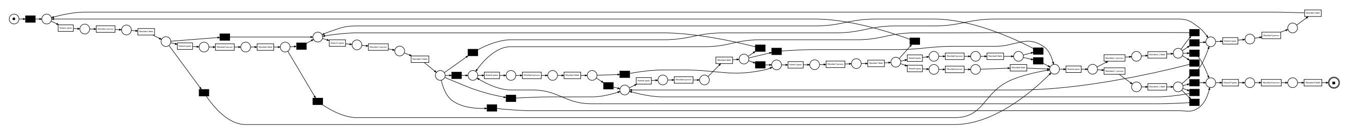
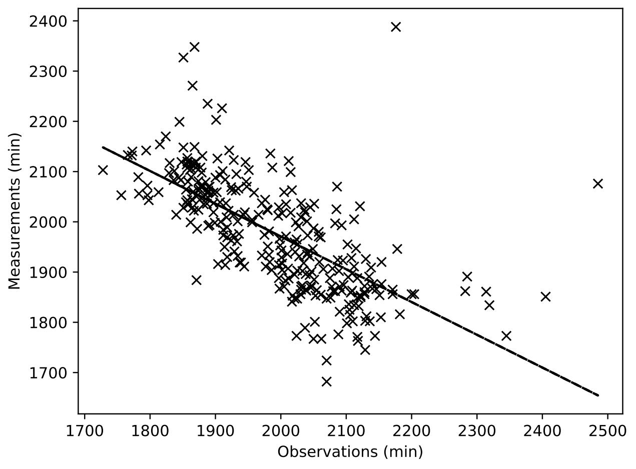
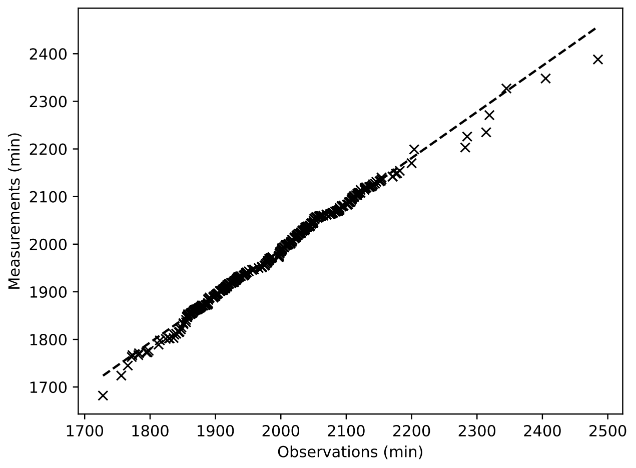


**Figure 8: Automatic generated process model of the wafer-fab simulation. The place and transitions are drawn as circles and rectangles respectively; the hidden transitions are drawn as solid rectangles.**



**(a) Comparing individual orders.**



**(b) Comparing orders sorted according to cycle time.**

**Figure 9: Comparing the observations and measurements for Order Type 1.**

routing process to the next machine. The *cycle time* of a wafer lot is the total time required by the wafer-fab to process the lot according to the recipe. The cycle time is used as the performance measurement of the wafer-fab as it is one of the KPIs in wafer-fab manufacturing.

## 5.2 Generated Model Validation

To validate the generated model, a wafer-fab simulation based on the case study is executed for 300 Order Type 1, 100 Order Type 2, and 3 respectively based on the frequency defined in the dataset. The events generated are aggregated in an event log, and the cycle time for all the orders is recorded. A process model is generated from the event log and simulated with the same orders as the original simulation model. Figure 8 shows the Petri-net model that is automatically generated from the whole event log. The cycle time of the orders in the process model is also recorded for comparison.

The observations (average cycle time) from the system are compared to the measurements of the generated model. Each point in Figure 9a shows the cycle time of the same order between the observations (x-axis) against the measurements (y-axis) for Order Type 1. A linear regression is performed between the observations and measurements, as shown in the solid line in Figure 9a. It can be seen from the regression line that there is a negative correlation between the observations and measurements. In addition, the coefficient of determination between the observations and measurements is 0.45. This means that for the individual orders, the measurements do not replicate the observations.

It is usually not feasible to determine if a model is absolutely valid over the complete domain of intended use or application [23]. Hence, the distribution of the system output is compared instead of individual order cycle time. The cycle time of the observations and measurements are sorted in ascending order before being plotted in Figure 9b. The observation for a point only matches the measurement in order, they do not represent the same order. It can be seen from the figure that there is a positive correlation between the observations and measurements. In addition, the coefficient of determination between the observations and measurements is 0.99. Hence, the overall distribution of cycle time by the generated model is similar to the output from the system.

The results are also similar for Order Types 2 and 3, which are not shown here due to space limitations. We also performed the two-sample Kolmogorov-Smirnov test between observations and measurements. The p-values obtained for each of the order types are 0.78, 0.81, and 0.21. Choosing a confidence level of 95%, the null hypothesis that the two distributions of the observations and measurements are identical cannot be rejected as the p-values are more than 0.05. These results also illustrated that standard sequential importance sampling algorithms are not applicable as the individual orders do not show the same output, only the overall distribution of the cycle time is similar.

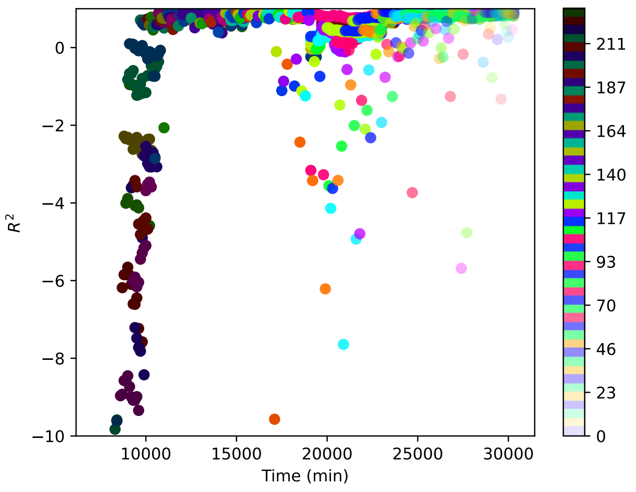


**Figure 10: Coefficient of determination of the samples over time. The color bars indicate the age of the samples, where the sample with age 0 is the newest sample generated.**

## 5.3 Identical Twin Experiment

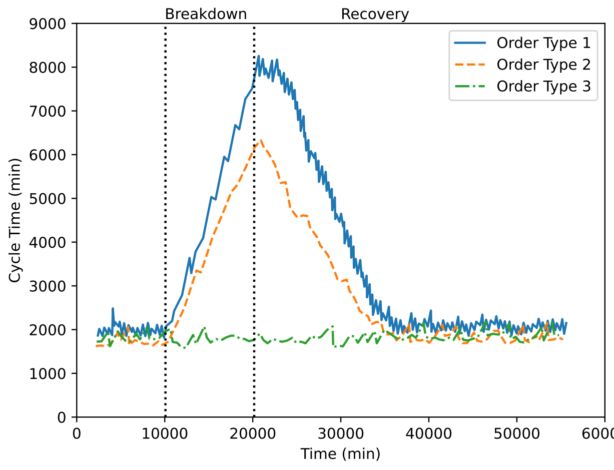
An identical twin experiment is conducted to evaluate the feasibility of the proposed data assimilation method. The wafer-fab simulation is executed for 21 days (30,240 minutes), with a warm-up period of 5 days (7,200 minutes). After the warm-up period, the new events and observations are recorded for the data assimilation process. The data assimilation is executed every 120 minutes, i.e, the sampling period $\Delta T = 120$. The time window $w$ is defined as 3 days (4,320 minutes). The number of samples $N_s$ is defined to be 10.

The fitness of the samples is evaluated by comparing the observations and measurements using the coefficient of determination. The coefficient of determination of the samples over time is plotted in Figure 10. The color represents the age of the sample, where the sample with age 0 is the newest sample generated. Darker colors indicate older samples while lighter colors indicate newer samples. As the data assimilation procedure proceeds with time, samples with lower $R^2$ are discarded and newer samples based on the windowed event logs are generated. Hence, the color of the sample becomes lighter as time progresses.

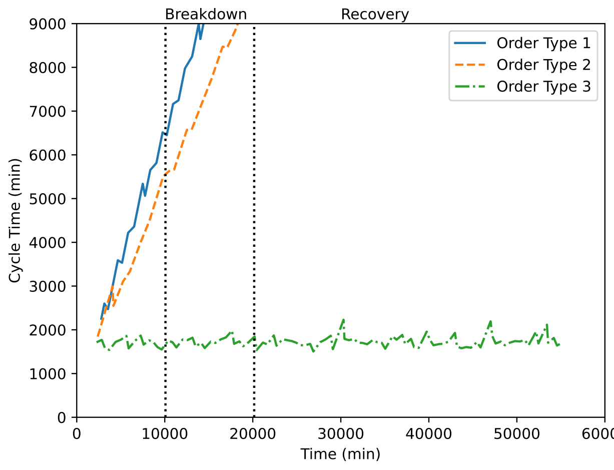
In Figure 10, the $R^2$ are very far from 1 at the beginning of the data assimilation. Due to insufficient events in the event log, the generated models are a poor estimate of the system. As the number of events aggregated in the event log increases, the $R^2$ of the newer samples improves, before converging towards 1. There are still some newer samples that are far from 1 due to different simulation states randomly initialized for each sample. However, the samples still generally converge towards 1, indicating that the newer samples have similar output compared to the system.

## 5.4 Data Assimilation for System Adaption

To evaluate the applicability of the data assimilation framework to handle the system model changes, we emulate structural changes in the original system. A time window of 7 days (10,080 minutes) is used in this experiment. As shown in Figure 7, `Station 4` is one of the bottlenecks in the wafer-fab factory which will impact Order Type 1 and 2. There are two machines for the `Station 4` – `Machine 4.1` and `Machine 4.2`. To emulate changes in the system, `Machine 4.1` will break down at 7th day (10,080 minutes). After breaking down, it will no longer be able to process any more lots. The breakdown will cause a bottleneck to occur in the factory, increasing the overall cycle time. A recovery operation will be performed at 14th day (20,160 minutes) to increase the capacity of `Machine 4.2` in order to clear the backlog orders and reduce the cycle time. Figure 11a shows the cycle time for each order type over time. It can be seen from the figure that the cycle time of Order Type 1 and 2 increased from the 7th day due to the breakdown. The cycle time of Order Type 3 is not affected as it does not require `Station 4` for processing. After the recovery operation has been executed at the 14th day, this eventually clears the backlog of orders and reduces the cycle time.



**(a) Simulation output from the original system when `Machine 4.1` broke down and a recovery operation is performed.**



**(b) Simulation output from the generated model.**

**Figure 11: Cycle time of each order type over time.**

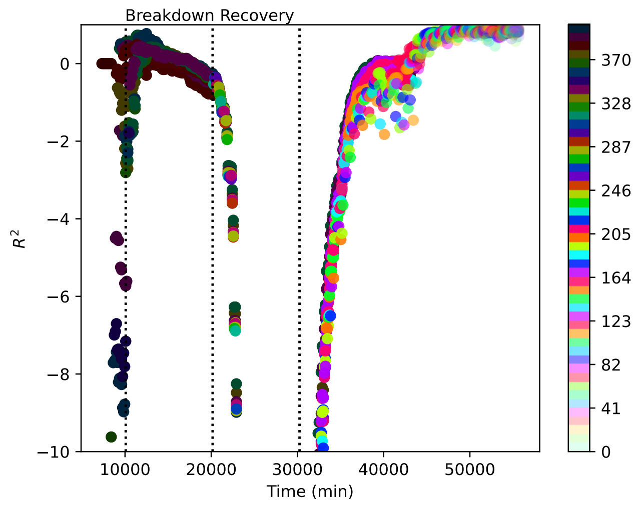


**Figure 12: Coefficient of determination of the samples over time after `Machine 4.1` broke down and a recovery operation is performed. The color bars indicate the age of the particles.**

Figure 11b shows the cycle time of each order type over time for one of the samples that are generated during the breakdown period. As `Machine 4.1` is already broken down, there are no events emitted from this machine; only events from `Machine 4.2` are aggregated. However, since the recovery operation has not been performed, the capacity of `Machine 4.2` is not increased yet. Hence, the generated model only extracted the original production capacity for `Machine 4.2`. The original capacity of `Machine 4.2` is insufficient to service all the incoming orders for `Station 4`, leading to increasing cycle time for Order Types 1 and 2, as seen in Figure 11b. The cycle time of Order Type 3 is not affected as it does not require `Station 4` for processing. Due to the increasing cycle time for Order Types 1 and 2, this will lead low coefficient of determination compared to the observations.

Similarly, the coefficient of determinations of the samples over time is plotted in Figure 12. Initially, the $R^2$ of the samples increases to a maximum $R^2 = 0.77$ at $12,480$ minutes. This indicates that the data assimilation is eventually able to generate models that are similar to the pre-breakdown system. However, the $R^2$ eventually decreases over time after $12,480$ minutes due to the large divergence of the generated models compared to the post-breakdown system.

There is a set of newer samples with increasing $R^2$ in Figure 12 after $30,240$ minutes. After the recovery and a time window of 7 days ($10,080$ minutes), sufficient events are aggregated in the event logs such that a new set of samples that can represent the post-recovery system is generated. The coefficient determination eventually converges to the ideal value of 1 as better-estimated models of the post-recovery system are generated. Hence, the proposed data assimilation framework can adaptively handle the changes in the system over time.

## 6 CONCLUSION

In conclusion, we propose an automatic model generation and data assimilation framework for modeling a smart manufacturing system. Automatic component-based model generation utilizes data and process mining techniques to generate a process model representing a smart manufacturing system. The events are observed from the real system and aggregated into an event log. Process discovery extracts the process model from the event log. The process model is enhanced with additional features mined from the event log to achieve a more realistic model that can be simulated as a discrete event simulation.

A new data assimilation method is also proposed to handle structural changes in the system by continuously generating new models and selecting models with the highest fitness. The events are used to automatically generate the corresponding process model for each sample. The system performance is also observed and aggregated during the data assimilation. The historical observations and measurements are compared to evaluate the fitness of the samples.

A case study using a wafer-fab simulation is used to evaluate the feasibility of the proposed framework. Validation of the automatically generated model shows that the generated model can only have statistically similar system output as the original system. The identical twin experiment shows the effectiveness of the proposed data assimilation method. Breakdown and recovery operations are emulated in the system to show the adaptability of the data assimilation framework to handle structural changes in the original system.

In future work, uncertainties and noise in the events and observations from the real system will be investigated to determine the robustness of the automatic model generation. The model generation can be improved by investigating techniques to discover different dispatching rules in the manufacturing system. The effectiveness of the data assimilation can also be improved by dynamically adjusting the sampling period. This will enable the framework to generate a better model through the aggregation of sufficient events. In addition, determining the appropriate number of samples will be addressed in future research. As introduced by [22], a system is described by its logic and input. Our proposed method is able to generate models that are validated by its logic; the input to the simulation is not captured in the model. Future research will consider modeling the input to the simulation too.

## ACKNOWLEDGMENTS

This work was supported by the A*STAR Cyber-Physical Production System (CPPS) – Towards Contextual and Intelligent Response Research Program, under the RIE2020 IAF-PP Grant A19C1a0018, Model Factory@SIMTech, and Ministry of Education (MOE) Singapore, Grant No.RG12/21 MoE Tier 1.

## REFERENCES

[1] Fan Bai, Feng Gu, Xiaolin Hu, and Song Guo. 2016. Particle Routing in Distributed Particle Filters for Large-Scale Spatial Temporal Systems. *IEEE Transactions on Parallel and Distributed Systems* 27, 2 (Feb. 2016), 481–493. https://doi.org/10.1109/TPDS.2015.2405912
[2] Sören Bergmann and Steffen Strassburger. 2010. Challenges for the Automatic Generation of Simulation Models for Production Systems. In *Proceedings of the 2010 Summer Computer Simulation Conference.* Society for Computer Simulation International, Ottawa, Ontario, Canada, 545–549.

[3] F Bouttier and P Courtier. 2002. Data assimilation concepts and methods March 1999. *Meteorological training course lecture series. ECMWF* 718 (2002), 59.

[4] Frank Chance, Jennifer Robinson, and John W. Fowler. 1996. Supporting Manufacturing with Simulation: Model Design, Development, and Deployment. In *Proceedings of the 28th Conference on Winter Simulation* (Coronado, California, USA) *(WSC '96)*. IEEE Computer Society, USA, 114–121. https://doi.org/10.1145/256562.256586

[5] Petar M Djuric, Jayesh H Kotecha, Jianqui Zhang, Yufei Huang, Tadesse Ghirmai, Mónica F Bugallo, and Joaquin Miguez. 2003. Particle filtering. *IEEE signal processing magazine* 20, 5 (2003), 19–38.

[6] Jonas Friederich, Giovanni Lugaresi, Sanja Lazarova-Molnar, and Andrea Matta. 2022. Process Mining for Dynamic Modeling of Smart Manufacturing Systems: Data Requirements. *Procedia CIRP* 107 (2022), 546–551.

[7] Anahita Farhang Ghahfarokhi, Gyunam Park, Alessandro Berti, and Wil M. P. van der Aalst. 2021. OCEL: A Standard for Object-Centric Event Logs. In *New Trends in Database and Information Systems*. Springer International Publishing, Cham, 169–175.

[8] S. Gillijns, O.B. Mendoza, J. Chandrasekar, B.L.R. De Moor, D.S. Bernstein, and A. Ridley. 2006. What is the ensemble Kalman filter and how well does it work?. In *2006 American Control Conference*. IEEE, Minneapolis, MN, USA, 6 pp.–. https://doi.org/10.1109/ACC.2006.1657419

[9] Xiaolin Hu and Peisheng Wu. 2019. A Data Assimilation Framework for Discrete Event Simulations. *ACM Transactions on Modeling and Computer Simulation* 29, 3 (June 2019), 17:1–17:26. https://doi.org/10.1145/3301502

[10] Yilin Huang, Mamadou D. Seck, and Alexander Verbraeck. 2011. From Data to Simulation Models: Component-based Model Generation with a Data-Driven Approach. In *Proceedings of the 2011 Winter Simulation Conference (WSC)*. IEEE, Phoenix, AZ, USA, 3719–3729. https://doi.org/10.1109/WSC.2011.6148065

[11] Nasser Jazdi. 2014. Cyber physical systems in the context of Industry 4.0. In *2014 IEEE international conference on automation, quality and testing, robotics*. IEEE, Cluj-Napoca, Romania, 1–4.

[12] David Kayton, Tim Teyner, Christopher Schwartz, and Reha Uzsoy. Fourth Quarter 1997. Focusing Maintenance Improvement Efforts in a Wafer Fabrication Facility Operating under the Theory of Constraints. *Production and Inventory Management Journal* 38, 4 (Fourth Quarter 1997), 51–57.

[13] D. Krenczyk, B. Skolud, and M. Olender. 2016. Semi-Automatic Simulation Model Generation of Virtual Dynamic Networks for Production Flow Planning. *IOP Conference Series: Materials Science and Engineering* 145, 4 (Aug. 2016), 042021. https://doi.org/10.1088/1757-899X/145/4/042021

[14] Sander JJ Leemans, Erik Poppe, and Moe T Wynn. 2019. Directly follows-based process mining: Exploration & a case study. In *2019 International Conference on Process Mining (ICPM)*. IEEE, Aachen, Germany, 25–32.

[15] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. 2013. Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. In *Application and Theory of Petri Nets and Concurrency*. Vol. 7927. Springer Berlin Heidelberg, Berlin, Heidelberg, 311–329. https://doi.org/10.1007/978-3-642-38697-8_17

[16] Giovanni Lugaresi. 2021. *Automated Generation and Exploitation of Discrete Event Simulation Models for Decision Making in Manufacturing*. Ph. D. Dissertation. Politecnico di Milano, Milan, Italy.

[17] Giovanni Lugaresi, Gianluca Aglio, Federico Folgheraiter, and Andrea Matta. 2019. Real-time validation of digital models for manufacturing systems: A novel signal-processing-based approach. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE, Vancouver, BC, Canada, 450–455.

[18] Giovanni Lugaresi, Sofia Gangemi, Giulia Gazzoni, and Andrea Matta. 2022. Online Validation of Simulation-Based Digital Twins Exploiting Time Series Analysis. In *2022 Winter Simulation Conference (WSC)*. IEEE, Singapore, 2912–2923.

[19] Alexander Mages, Carina Mieth, Jens Hetzler, Fadil Kallat, Jakob Rehof, Christian Riest, and Tristan Schäfer. 2022. Automatic Component-Based Synthesis of User-Configured Manufacturing Simulation Models. In *2022 Winter Simulation Conference (WSC)*. IEEE, Singapore, 1841–1852.

[20] Lars Mönch, John W Fowler, and Scott J Mason. 2012. *Production planning and control for semiconductor wafer fabrication facilities: modeling, analysis, and systems*. Vol. 52. Springer Science & Business Media, New York, NY, USA.

[21] Lucy E Morgan and Russell R Barton. 2022. Fourier trajectory analysis for system discrimination. *European Journal of Operational Research* 296, 1 (2022), 203–217.

[22] B Nelson et al. 2013. Foundations and methods of stochastic simulation. *A first course. International series in operations research & management science* 187 (2013), 313 pages.

[23] Robert G Sargent. 2010. Verification and validation of simulation models. In *Proceedings of the 2010 winter simulation conference*. IEEE, Baltimore, MD, USA, 166–183.

[24] Denise Maria Vecino Sato, Sheila Cristiana De Freitas, Jean Paul Barddal, and Edson Emilio Scalabrin. 2021. A Survey on Concept Drift in Process Mining. *Comput. Surveys* 54, 9 (Oct. 2021), 189:1–189:38. https://doi.org/10.1145/3472752

[25] Moon GI Seok, Chew Wye Chan, Wentong Cai, Hessam S. Sarjoughian, and Daejin Park. 2020. Runtime Abstraction-Level Conversion of Discrete-Event Wafer-fabrication Models for Simulation Acceleration. In *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '20)*. Association for Computing Machinery, New York, NY, USA, 83–92. https://doi.org/10.1145/3384441.3395982

[26] Young Jun Son and Richard A Wysk. 2001. Automatic Simulation Model Generation for Simulation-Based, Real-Time Shop Floor Control. *Computers in Industry* 45, 3 (July 2001), 291–308. https://doi.org/10.1016/S0166-3615(01)00086-0

[27] Wil MP van der Aalst. 2010. Process discovery: Capturing the invisible. *IEEE Computational Intelligence Magazine* 5, 1 (2010), 28–41.

[28] Harry L. Van Trees and Kristine L. Bell. 2007. *A Tutorial on Particle Filters for Online Nonlinear/NonGaussian Bayesian Tracking*. Wiley-IEEE press, NY, USA, 723–737. https://doi.org/10.1109/9780470544198.ch73

[29] A. J. M. M. Weijters, Wil M.P. van der Aalst, and Ana K. A. de Medeiros. 2006. *Process mining with the HeuristicsMiner algorithm*. Technical Report. Technische Universiteit Eindhoven, Eindhoven, Netherlands.

[30] Xu Xie and Alexander Verbraeck. 2019. A Particle Filter-Based Data Assimilation Framework for Discrete Event Simulations. *SIMULATION* 95, 11 (Nov. 2019), 1027–1053. https://doi.org/10.1177/0037549718798466

[31] Xu Xie, Alexander Verbraeck, and Feng Gu. 2016. Data Assimilation in Discrete Event Simulations - a Rollback Based Sequential Monte Carlo Approach. In *2016 Symposium on Theory of Modeling and Simulation (TMS-DEVS)*. IEEE, Pasadena, CA, 1–8. https://doi.org/10.23919/TMS.2016.7918817