IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 8 November 2022, accepted 21 November 2022, date of publication 28 November 2022,
date of current version 5 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3225399

APPLIED RESEARCH

Adaptive Model Verification for Modularized Industry 4.0 Applications

XIN XIN[1,2], SYE LOONG KEOH[1], (Member, IEEE), MICHELE SEVEGNANI[1], MARTIN SAERBECK[2], AND TECK PING KHOO[2]
[1]School of Computing Science, University of Glasgow, G12 8RZ Glasgow, U.K.
[2]Digital Service, TÜV SÜD Asia Pacific, Singapore 609937

Corresponding author: Xin Xin (x.xin.2@research.gla.ac.uk)

The work of Xin Xin was supported in part by the Singapore Economic Development Board (EDB) through the Industrial Postgraduate Programme (IPP) Grant. The work of Michele Sevegnani was supported in part by the Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/F033206/1, in part by the Formal Methods for Agritech Resilience Modelling (FARM) under Grant EP/S035362/1, in part by the Multi-Perspective Design of IoT Cybersecurity in Ground and Aerial Vehicles (MAGIC), and in part by the Amazon Research Award.

ABSTRACT Cyber-Physical Systems (CPSs) are the core of Industry 4.0 applications, integrating advanced technologies such as sensing, data analytics, and artificial intelligence. This kind of combination typically consists of networked sensors and decision-making processes in which sensor-generated data drive the control decisions. Hence, the trustworthiness of the sensors is essential to guarantee performance, safety and quality during operation. Formal model verification techniques are a valuable tool allowing strong reasoning about the high-level design of CPSs. However, the uncertainty exhibited by the underlying sensor networks is often ignored. Manufacturing processes typically involve composition of various modular CPSs that work as a whole, such as multiple Collaborative Robots (cobots) working together as a production line, which improves the flexibility and resilience of the production process. It is still challenging to verify this class of compositional process while also considering uncertainty. We propose a novel verification framework for modular CPSs that combines sensor-level data-driven fault detection and system-level model-driven probabilistic model checking. The resulting framework can rigorously quantify sensor readings' trustworthiness, enabling formal reasoning for system failure prediction and reliability analysis. We validated our approach on a cobots-based manufacturing process.

INDEX TERMS Probabilistic model checking, sensor networks, trustworthiness, cyber-physical system, collaborative robot, industry 4.0.

I. INTRODUCTION

Cyber-Physical Systems (CPSs) connecting physical devices into a cyber-network are emerging as an efficient Industry 4.0 paradigm to enable many industrial manufacturing use cases [1], [2], [3]. Industry 4.0 aims to integrate operational technologies (OT) and information technologies (IT) to connect industrial assets, including machines and control systems, with the information systems and business processes to quickly and dynamically respond to demand changes [4], [5], [6]. This has accelerated the adoption of collaborative robots (cobots) working together on production lines. CPSs provide the capability of decentralisation, modularity and interoperability to enable advanced industrial automation.

With the utilisation of Artificial Intelligence (AI), machine learning and data analytics, operators are now able to collect large volumes of data from operational systems and equipment at run time. Subsequently, computers can simulate human decisions with complex algorithms to derive insights and enable self-control actions. Such an integration significantly accelerates the automation processes to improve the operation quality, ensure process safety and predict potential failures. Although all of these technologies are connected, there are still challenges that have not been addressed in real-world deployments. Firstly, the system model used to define the operational behaviour of the system

The associate editor coordinating the review of this manuscript and approving it for publication was Agostino Forestiero.

is typically static and does not accurately reflect the actual behaviour of the equipment or sensors forming the CPS over time. It is thus important to identify any deviation from the expected behaviour to reduce equipment down time and to effectively plan for component replacement as well as to ensure compliance with safety regulations. Secondly, there is a strong dependency between the accurate prediction of an equipment's behaviour and the data collected from the sensors instrumenting the equipment. This means that it is extremely important to acquire *trustworthy* sensor readings to ensure the accurate prediction of the system behaviour. Most of the systems deployed on the field currently assume that the acquired sensor readings are accurate, which may lead to inaccuracy in their predictive algorithms. However, the uncertainty exhibited by the underlying sensor networks is difficult to determine while the sensor network is deployed. For instance, unreliable connectivity or hardware failures is one of the most common issues, which results in intermittent readings. Another common issue is the environment being out of range of the transducer due to manufacturing process changes, e.g., the environment light is too bright for the camera sensor compared to the initial deployment condition. We advocate the idea that all decisions made are only as good as the sensor data that they are based on. Lastly, CPSs typically involve composition of various sensor networks, e.g., cobots and Automated Guided Vehicles (AGVs) working together. This means that the system can be flexibly formed with multiple independent functional components, thus achieving modularity. Although each component's behaviour can be verified independently through formal verification, it is challenging to reflect on their behaviour when plugged in as a modular component at the system level.

In this paper, we propose a component-based run-time model verification approach for Industry 4.0 applications. Two different model cobots are used to evaluate this approach. Firstly, these two cobots are verified independently based on our previous work [7]. Subsequently, the verified two cobot models can be dynamically composed into a manufacturing process. Afterwards, this manufacturing process is abstracted as a system-level model that can be verified at run time using probabilistic model checking techniques. We built on our previous work on an approach to determine the *trustworthiness* of sensor readings at run time. Subsequently, this quantified *trustworthiness* value is fed back as sensor networks' *confidence score* to reflect the impact at model level though probabilistic model checking, this allows for the accurate prediction of failures. The contributions of this paper are as follows:

- A new method to create a component-based system probabilistic model including interacting sub-modules as child models and higher level system abstraction. This method allows independently verified child models (e.g., individual cobots) to form a system-level probabilistic model according to the changing requirements of Industry 4.0 applications that can be verified at a higher level.
- Integrating data-driven models that are based on quantified *trustworthiness* of the sensor readings with a probabilistic model to enable run-time verification of the system behaviour. With this, a more accurate reflection of the system behaviour at run-time can be achieved, therefore enabling the prediction of failures more accurately.

This paper is organised as follows: Section II provides the background of this research and related work. Section III describes the proposed run-time compositional probabilistic model checking design, with the ability to verify a CPS consisting of multiple cobots at run time. Section IV presents an implementation to predict the failure probability of a painting system using two cobots. The experiment and results are presented in Section V. Section VI provides further insights on the experiment results. Finally, we conclude the paper with future work in Section VII.

II. RELATED WORK

A. CPS VERIFICATION

Model checking is widely used to verify modern system behaviour and analyse system reliability. Calder and Sevegnani [8] introduced a stochastic probabilistic model checking framework for failure prediction of a critical communication system. This framework is based on a discrete space model and temporal logic to predict likelihood of service failure within a given time bounds, and quantify the impact of lower level components on service availability. However, the proposed framework is difficult to model run time behaviours due to the unreliable readings from sensors. Filieri et al. [9] introduced a run-time probabilistic model checking approach to evaluate the satisfaction of reliability requirements at run time. The authors defined two phases, namely design-time phase and run-time phase. At design-time, a Discrete-Time Markov Chain (DTMC) model is pre-computed, and a set of symbolic expressions is defined to represent satisfaction of the requirements. As this model transition values are known only at run time and may change over time, a set of variables are used to represent the transition probabilities. Subsequently, the verification is performed at run time by replacing the transition variables with the real values gathered by a monitoring system. However, the performance of this approach is not only the DTMC model itself, but also depends on the monitoring system inputs which is hard to apply to more general scenarios. Li et al. [10] presented a dynamic adaptation probabilistic model checker approach to improve self-adaptive systems' utility. They define a Markov Decision Process (MDP) [11] model as the system abstraction and operator provides initial transition parameters according to experiment results and experience. Subsequently, the transition parameters are updated onto the MDP model according to the operation parameters at run time. Over time, the MDP model adapts itself with the self-adaptive system's behaviour. This approach relies on the operator's actions and the effect to the system model can be accurately measured. Epifani et al. [12] proposed another novel dynamic probabilistic model checking framework based on KAMI (Keep Alive Models with Implementations). This framework is based on the Bayesian Estimation Theory (BET) to estimate the transition matrix according to the run-time system. Subsequently, the estimated transition matrix is applied to a DTMC model to increase the failure prediction accuracy. Even so, to quantify the run-time variables is still a challenge, e.g., system embedded sensors' trustworthiness.

Considering the complexity and tight interactions of CPSs, Statistical Model Checker (SMC) is proposed [13], [14] to tackle two obstacles [15] of modern CPSs. SMC is a simulation-based approach to sample the behaviours and check conformance to the temporal formula. Younes et al. [16] compared two probabilistic model checking techniques, Numerical- and Statistical-probabilistic model checking. The result showed that both techniques have similar performance, but the statistical approach scales better with the size of the state space and requires less memory. Zarei et al. [17] proposed another SMC approach to verify learning-based CPSs. This kind of CPSs employs machine learning algorithm-based controllers, e.g., Neural Network, that increases complexity and non-linearity. Traditional verification techniques face state-space search and scalability challenges. Thus, they built the SMC based on the Clopper-Pearson confidence levels and defined specifications using Signal Temporal Logic (STL) to verify the reachability, safety and performance. The results showed that it is feasible to use statistical verification for learning-based CPSs. Even so, determining the CPS's run-time characteristics is still a challenge as the sensor readings are unreliable which will significantly affect the reliability of the verification.

Apart from the SMC approach, *ModelPlex* [18] provides correctness guarantees for CPSs at run time. It combines *Model Monitor* to check the previous state and current state for compliance with the model, *Controller Monitor* checks the output of a controller implementation against the controller model, while *Prediction Monitor* checks the impact of deviation from the model to predict the eventual state that might cause failures. *ModelPlex* is based on *differential dynamic logic* **dL** [19] and has been applied in robotic applications [20], [21], [22] using the tool KeYmaera X [23], [24]. However, the run-time trustworthiness of the sensors' readings are not reflected through the *Controller monitor* and *Prediction monitor*.

Together with model checking, temporal logic is a popular formalism language for specifying reactive system behaviours. Traditionally, temporal logic language has been used for formal verification, such as LTL to capture safety and reachability requirements over Boolean predicates defined over the state space. Computation Tree Logic (CTL) allows the expression of requirements over all computations branching from a given state [25]. Kamide and Yano [26] introduced a sequential Linear Temporal Logic (sLTL) and a sequential Computation Tree Logic (sCTL) by extending LTL and CTL to represent hierarchical information and structures. This research work defined the translations from sLTL and sCTL into LTL and CTL to verify hierarchical systems by reusing the standard LTL- and CTL-based model checking algorithms.

Over the last few years, the Robot Operating System (ROS) has become a popular software framework for distributed robotics and CPSs. A ROS-based Run-time Verification (ROSRV) framework is an approach that incorporates a middle layer to intercept messages in order to verify the run-time system behaviour [27], [28]. The ROSRV provides a functional layer to intercept all messages between the slave layers to master layers, and by understanding the communication between them, the system is able to enforce the desired system behaviour based on the safety policies. However, such verification systems only results in the system conforming to the behaviour and safety policies, but it has no ability to predict failures in advance. Furthermore, the middle layer is actually incurring overheads and it can become a bottleneck when there are large number of messages being exchanged in a large scale deployment.

Ferrando et al. [29] introduced another ROS-based runtime verification framework, ROSMonitoring. This framework automatically verifies messages against formally specified properties by adding a monitor through ROS node instrumentation instead of creating a middle layer. It provides the flexibility to scale up and to choose the specification formalism, such as Linear Temporal Logic (LTL) or Signal Temporal Logic (STL). However, the ROSMonitoring approach can only be applied to ROS-based CPSs.

To describe and analyse distributed systems rigorously, Paul and Nancy [30] presented a Dynamic Input/Output Automata (DIOA) that allows create and destroy components dynamically. Civit and Potop-Butucaru [31] extended DIOA to a probabilistic framework to model a dynamic probabilistic systems, for instance an industry 4.0 application using multiple CPSs to work on one manufacturing process. However, the DIOA models analyse a system only. It lacks an approach to verify such dynamic systems.

B. SENSOR FAULT DETECTION

The current research works tend to assume that the sensor readings are accurate and reliable of modern CPSs. However, a sensor might behave with uncertainties in real-world deployments due to complex environmental conditions or incidentally placement, such as unreliable connectivity, irregular power supplies or calibration errors. In order to analyse sensor reliability, three sensor fault categories are systematically classified by Ni et al. [32]. This classification is defined according to the nature of sensor attributes, which are *environment features*, *system features or specifications*, and *data features*. Furthermore, these categories are widely employed in sensor fault detection algorithms. Sharma et al. [33] proposed four methods to detect the above sensor faults, which are rule-based, time-series analysis-based, learning-based and estimation methods. Apart from statistical approaches, Park et al. [34] proposed a data-driven light-weight real-time sensor fault detection system.

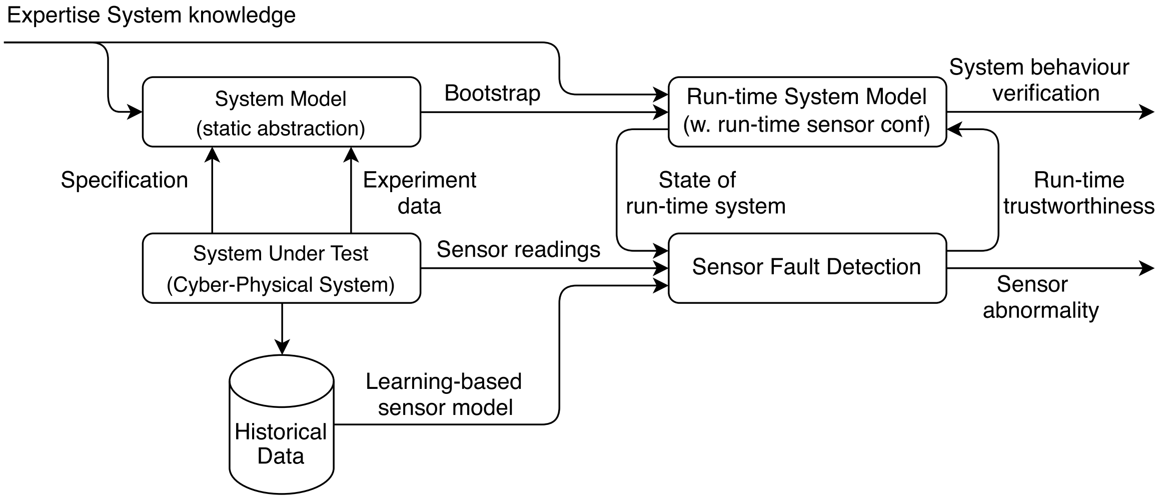

FIGURE 1. Architecture of run-time model verification.

This system employs a Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) deep learning model to detect sensor faults. It overcomes the limitation of a pure mathematical approach, which is sensitive to noise and system complexity. However, the quality of machine learning-based techniques depends on the dataset that covers a wide variety of use cases. To get high-quality training data sets is always the challenge in sensor network-based applications. Li et al. [35] summarised recent advances in sensor fault diagnosis. The authors classified sensor faults into two categories: incipient failure and abrupt failure. Together with three fault diagnosis techniques: model-based approaches, knowledge-based approaches and data-driven approaches, the sensor reliability can be verified and quantified.

In this research, we extend the existing research works to embed quantified sensor *trustworthiness* into a system-level model to analyse the reliability of the CPS. Moreover, our approach provides a practical methodology to quantify the sensor reading's *trustworthiness* to enable accurate system impact analysis at run time. In addition, a real test-bed environment consisting of painting operation by cobots was set up to validate our approach.

III. RUN-TIME COMPOSITIONAL MODEL VERIFICATION

We present a run-time compositional verification framework that combines data- and model-driven approaches to analyse the reliability of CPSs. Fig. 1 illustrates the architecture of the proposed framework.

First, a formal system model is defined according to the specification of the CPS, expert knowledge and experiment data. This system model represents a static system-level abstraction that is required to bootstrap the model verification at run time. It also specifies the initial state transition probability matrix of the system to reflect the overall system behaviour. The sensor fault detection module then continuously learns about the sensor behaviour and quantifies the sensor's *trustworthiness* against expectation. A set of sensor data is first collected under the operator supervision to ensure the system works as expected. Subsequently, this dataset is used to profile the sensor as its *normal behaviour* using time series analysis, estimation and rule-based methods following data-driven approach. If any deviation from the *normal behaviour profile* is detected, the quantified sensor trustworthiness will be updated at run time, this is termed as the *sensor confidence*. Subsequently, the confidence score is fed into the system model to update the transition probabilistic matrix to form a run-time system model to reflect the dynamic nature of a CPS. For instance, if a sensor fault is detected, the transition probability matrix of the system model is updated with lower *confidence* in the trustworthiness of the sensor data, which will in turn increase the probability of transition from a working state to an error state, thus leading to a system failure. With this, the system model evolves over time through this process continuously, taking into consideration the trustworthiness of run-time sensor readings to derive the appropriate probability of state transitions.

The proposed framework also supports modularity in that the top-level system model can be decomposed into lower-level child models to provide flexibility to verify the system at different levels of abstraction. Each child model can be verified independently through traditional model checking during the design phase. Subsequently, the proposed approach updates the transition probability matrix at run time. With this, it is now feasible for the verified child model to be reused and composed dynamically to form a higher level CPS. Fig. 2 shows how the child model is defined with interfaces of input and output states such that they can be integrated with each other or with a higher level system model easily.

A. RUN-TIME PROBABILISTIC SYSTEM MODEL

In line with a traditional formal model, the system states and transitions are defined according to the system specification,

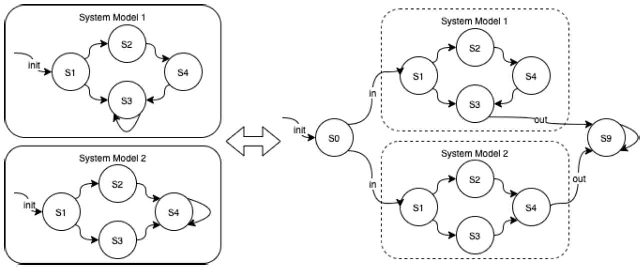

FIGURE 2. Structure of compositional model verification.

which means that they will not change during the operation phase. In a real-life scenario, the transition probability matrix will change over time due to sensor wear-and-tear and drift. In our run-time probabilistic model, we take into account the accuracy of the sensor reading, hence termed *trustworthiness* (a.k.a. *sensor confidence*), which will then be used to update the transition matrix of the system model dynamically.

In order to quantify the sensor's *trustworthiness*, our approach [7] builds a *Sensor Normal Behaviour* profile based on the historical data extracted from the sensors. This includes sensors' run-time *statistical characteristics*, *estimated reading range* and *drift trend*. In essence, the profile is based on the number of readings received, the mean value and the standard deviation of each working state. In the operational phase, the sensor's run-time readings are then compared against the *Sensor Normal Behaviour* using a rule-based engine, subsequently deriving a *Sensor Confidence Score* to be fed into the probabilistic model to update the transition probability matrix dynamically. A run-time probabilistic model, M_{rt} is defined as follows:

$$M_{rt} = (S, s_{init}, P_{runtime}, L) \tag{1}$$

where S is a finite set of machine states of the system, $s_{init} \in S$ is the initial state, $P_{runtime} : S \times S \times T \rightarrow [0, 1]$ is the run-time transition probability matrix that the transition keeps updating over the finite time period $T \subset \{0, \dots, n\}$, where $\sum_{s' \in S} P(s, s', t) = 1$ for all $s \in S, t \in T$, and $L : S \rightarrow 2^{AP}$ are function-labelling states with atomic propositions.

B. CHILD MODEL

A child model is an abstraction of a minimum fully functional system in a production process, e.g., a turn-mill machine, a reaction vessel, or a cobot. Normally, it is defined during the design phase to verify all system behaviours that satisfy the specification. In this paper, we define a child model, M_{child} by extending the run-time probabilistic model as a six-tuple to provide the capability to be integrated with other models to form a larger system model.

$$M_{child} = (S, s_{init}, P_{child}, L, S_{in}, S_{out}) \tag{2}$$

where S, s_{init}, P_{child} and L are defined as the run-time probabilistic model M_{rt} and two new elements, S_{in} and S_{out} are added. $S_{in} \subset S$ is the entry state of the model, while $S_{out} \subset S$ is the exit state of the model.

C. SYSTEM-LEVEL MODEL

Multiple child models can be added to form a system-level model at run time and this fits well with Industry 4.0 applications as the manufacturing process is formed dynamically by multiple fully functional sub-systems with a centralised control system. For instance, two collaborative robots form a sequential painting process in a production line in which one cobot moves the workpiece to the designated zone, while the other cobot performs the painting task. In this context, the cobot is defined as a child model and it is used as part of the system-level manufacturing process defined as:

$$M_{system} = (S, s_{init}, P_{system}, L, M_{child_1} || \dots || M_{child_k}) \tag{3}$$

where M_{system} is the top level system model, S, s_{init}, P_{system} and L are defined as in M_{rt}, while $M_{child_1} || \dots || M_{child_k}$ is a set of the child models (Eq. 2) representing the sub-system of the top level system.

Consider the relationship between the child model and the system model, the following constraints should be satisfied:

1) $S_{system} \cap S_{child_i} = S_{in_i} \cup S_{out_i} \quad \forall\, i \in \{1, \dots, k\}$
2) $S_{in_i} \cap S_{out_i} = \emptyset \quad \forall\, i \in \{1, \dots, k\}$
3) $S_{child_i} \cap S_{child_j} = \emptyset \quad \forall i \neq j$
4) $P_{system}(s, s', t) = 0$
   $\forall\, s \in S_{in_i}, s' \in S_{system},\ t \in T \textit{ and } i \in \{1, \dots, k\}$
5) $P_{system}(s, s', t) = 0$
   $\forall\, s \in S_{system},\ s' \in S_{out_i},\ t \in T \textit{ and } i \in \{1, \dots, k\}$

The conjunction of system model S_{system} and child models S_{child_i} must only contain interface-states S_{in_i} and S_{out_i} as illustrated in constraint 1. Secondly, there cannot be any overlap states between S_{in_i} and S_{out_i} (constraint 2). Similarly for all the child models, there cannot be any overlap states between them (constraint 3). Lastly, constraint 4 and 5 do not allow S_{system} transition, that is from S_{in_i} states, and to S_{out_i} states at any time $t \in T$, and $i \in \{1, \dots, k\}$, respectively.

At the top level of the system model, there should not have input state(s) and output state(s) to ensure finite states of the system.

Semantics of the proposed approach are stated as follows:

1) $S = S_{system} \cup S_{child_i} \quad \forall\, i \in \{1, \dots, k\}$
2) $P = [P_{system} \quad P_{child_1} || \dots || P_{child_k}]$
3) $P(s, s', t) = P_{system}(s, s', t)$
   $\forall s \in (S_{system}/S_{in_i}), s' \in S, t \in T, \textit{ and } i \in \{1, \dots, k\}$
4) $P(s, s', t) = P_{child_i}(s, s', t)$
   $\forall s \in (S_{child_i}/S_{out_i}), s' \in S, t \in T, \textit{ and } i \in \{1, \dots, k\}$

In order to verify the system as a whole, the global state space is represented as S, which is the union set of all the system model's states S_{system} and all child models' states S_{child_i} (semantic 1). Similarly, the global transition matrix P concatenates the transitions of both the system model and the child models (semantic 2). If the run-time state is a system-level state S_{system} and is not an interface S_{in_i} state, the system transition P_{system} is applied (semantic 3). If the run-time state is a child state S_{child} and is not an interface S_{out_i} state, the system transition P_{child} is applied (semantic 4).

With the semantics of the proposed approach and the five constraints above, the design guarantees that there

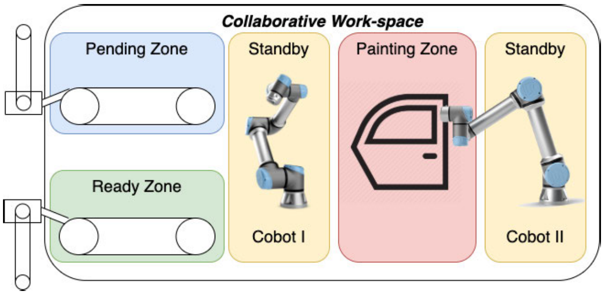

FIGURE 3. The collaborative workspace.

are no overlapped states between the system-level model and the child models except the interface-states. Therefore, this allows the high-level system and child modules to be modelled separately and verified as a whole system sequentially at run time.

D. TEMPORAL LOGIC PROPERTY QUERY

Once the system level model has been defined, it can be verified and then queried to predict the system's potential failure at run time. Computation Tree Logic (CTL) is used to evaluate the failure probability. The probability of system failure is expressed by the following Probabilistic Computation Tree Logic (PCTL) [36] formula:

$$P_{failure} =? \; [\, F^{\leq t} \, (S_{failure})]$$

where $P_{failure}$ is the system failure in the next t time. The failure state $S_{failure}$ is defined in the system model in Fig. 5.

IV. EXPERIMENT DESIGN

We designed an experiment using two cobot arms as a testbed to evaluate the proposed run-time compositional verification framework. These two cobots synchronise their actions automatically to complete the painting task of one workpiece.

Fig. 3 illustrates three working zones in the collaborative workspace. Cobot I is in charge of moving the workpiece from the *pending* zone to the *painting* zone, while Cobot II executes the painting work at the *painting* zone. When the painting work is completed, Cobot I must then move the workpiece to the *ready* zone. During the painting process, one cobot must be in the *standby* zone with standby position when another cobot is working. Each cobot is equipped with a depth-camera to detect the working zone and workpiece using image processing techniques.

The electrical current in Amperes (A) of the cobots is the key indicator to determine the machine's working condition. For instance, if the cobot is obstructed, the current reading will be out of the normal reading range. We monitored the current reading of the base joint, shoulder joint, elbow joint and wrist joint in this experiment first to build the cobot's *Sensor Normal Behaviour* profile and determine the *Sensor Confidence Score* in order to evaluate the proposed approach.

A. ASSUMPTIONS

In order to focus on the main actions and to simplify the painting process model, the following assumptions are defined according to the expertise:

- All cobots are working in the collaborative workspace, including *pending* zone, *ready* zone, *painting* zone and *standby* location.
- There is no shared space between *pending* zone, *ready* zone, *painting* zone and *standby* location.
- The three critical parameters, zone detection accuracy, workpiece detection accuracy and current readings of cobot's joints, can be queried through the painting system and cobot-system's Application Programming Interfaces (APIs).
- Five sensor fault types were used to compute the *Sensor Confidence Score* at run time. In addition, we assigned the following weight for each fault type based on the occurrence and severity.
  - Intermittent fault: 0.1
  - Stuck-at fault: 0.1
  - Spike fault: 0.1
  - Follow estimated reading range: 0.1
  - Sensor data pattern match: 0.6
- The servicing cycle is twenty days.
- The cobot's safe working range is defined according to ISO/TS 15066:2016. The *pressure* should be within 160 N/cm^2 of quasi-static contact and 2 N/cm^2 of transient contact. Additionally, the *force* should be within 210 N of quasi-static contact and 2 N of transient contact [37].

B. CHILD MODEL – COBOT ARM

As shown in Fig. 4, an operating cobot can be represented by eleven states to illustrate its working status.

S_1 *Standby* is the entry state in which a cobot is ready for the duty. In this state, the *force* and *pressure* should be in the safe working range.

S_2 *Idle* is the state that cobot prepares to move its arm from *standby* location to working zone, e.g., *painting* zone, *pending* zone. In this state, the *force* and *pressure* are kept in safe range. The cobot turns on the camera and triggers its image processing engine to determine the position of the workpiece.

S_3 *Zone detection* detects and confirms the target zone location. The cobot should move the gripper to a more precise position.

S_4 *Workpiece detection* detects and calculates the precise location and the size of workpiece. The cobot operates its gripper to grip and pick up the workpiece accordingly.

S_5 *Execute mission* means that the cobot performs the task as instructed.

S_6 *Controlled parking* is the state that the cobot reduces the Tool Centre Point (TCP) [38] speed safely, stop working and moves as commanded by the controller.

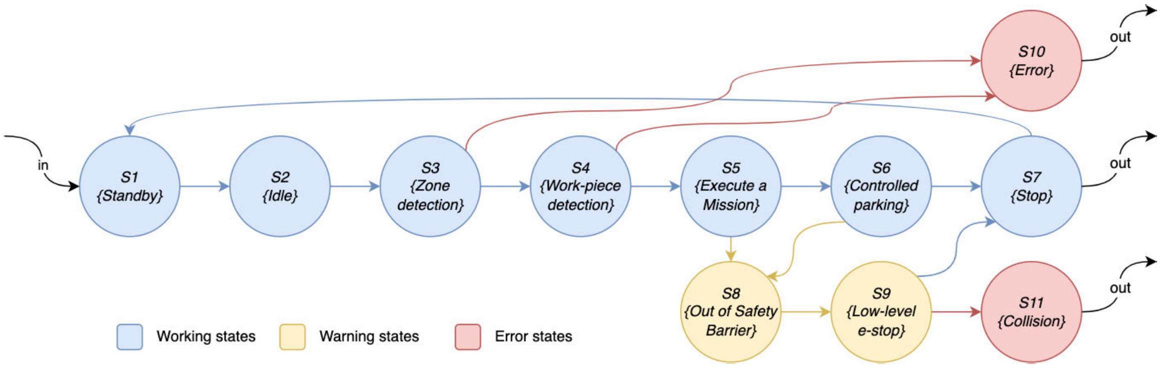

FIGURE 4. The cobot child model M_{cobot}.

In this state, the *force* and *pressure* are kept in the safe range.

S_7 *Stop* is an exit state in which the cobot is stationary and does not work for any purpose. In addition, the *force* and *pressure* are in safe range.

S_8 *Out of safety barrier* is the state that one or both *force* and *pressure* is out of the safe range. In this state, the cobot must stop working immediately.

S_9 *Low-level e-stop* means the cobot triggers the emergency-stop function.

S_{10} *Error* is a state indicating that there is a zone or workpiece detection error.

S_{11} *Collision* is a state that cobot stopped and one or both *force* and *pressure* is still out of the safe range.

In order to abstract the cobot system as a child model, we extend the DTMC as a 6-tuple as described in Section III. In this experiment, we use M_{cobot} to represents the child model M_{child}.

$$M_{cobot} = (S, s_{init}, P_{cobot}, L, S_{in}, S_{out})$$

where M_{cobot} is a DTMC model of the cobot system, S is the set of eleven cobot states, s_{init} is the initial state S_1. The initial transition matrix is defined according to the specification in the design phase. Subsequently, during the operation phase, the state transition matrix P_{cobot} (c.f. Eq. 4), shown at the bottom of the page, will be updated based on three factors according to the run-time condition, namely *zone detection accuracy*, *workpiece detection accuracy* and *sensor confidence score*.

The resulting model is used to verify the cobot system behaviour individually. With the sensor confidence score derived at run time to update the transition matrix, P_{cobot}, it will accurately reflect the behaviour of the cobot at the operational phase.

C. SYSTEM MODEL - PAINTING SYSTEM

The system model is an abstraction of the painting process that includes five machine states, and is composed of two instances of the cobot model as the child models. In particular, Cobot I is responsible for moving the workpiece between the pending zone, painting zone, and ready zone; while Cobot II paints the workpiece in the painting zone only. In this system, both cobots should not execute their tasks concurrently, in order to avoid a potential collision. For example, while Cobot I is moving a workpiece to the painting zone, Cobot II should be stationary at the standby location with a safe standby pose.

Fig. 5 illustrates the system model that defines the five system states and two child cobot models,

S_1 *Idle* is the initial state of the painting process, the system checks all the modules' status and waits for the task to be executed.

S_2 *Plan* is the state that the system retrieves the task details, plans the steps and adjusts running parameters.

$$P_{cobot} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & zone_{acu} & 0 & 0 & 0 & 0 & 0 & 1 - zone_{acu} & 0 \\ 0 & 0 & 0 & 0 & wp_{acu} & 0 & 0 & 0 & 0 & 1 - wp_{acu} & 0 \\ 0 & 0 & 0 & 0 & 0 & sn_{conf} & 0 & 0 & 1 - sn_{conf} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & sn_{conf} & 0 & 1 - sn_{conf} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & sn_{conf} & 0 & 0 & 0 & 0 & 1 - sn_{conf} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4}$$

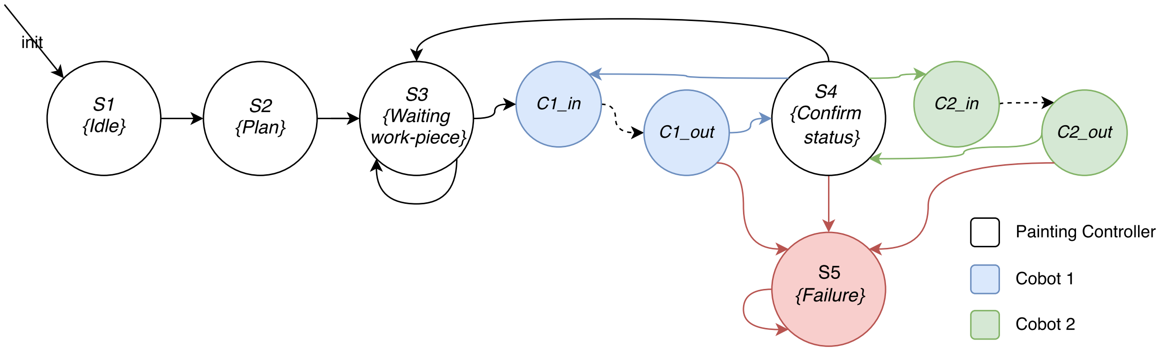

FIGURE 5. The painting system model.

- S_3 *Waiting workpiece* is a state that keeps detecting the workpiece in the pending zone. If there is a workpiece detected, the system should trigger the planned action.
- S_4 *Confirm status* is a state that ensures all modules are ready to move to the next step. We assume that there may be a random error with this state leading to the transition to S_5 *Failure* state, since the painting system checks with cobot systems and process control modules at this stage. According to the empirical rule and statistical studies of industrial processes [39], we assume it is a quarterly failure event, which approximates to 0.9875 ($\mu \pm 2.5\sigma$).
- S_5 *Failure* means the painting system is working with an unexpected behaviour that may cause subsequent hazard or injury.
- C_1 $C1_{in}$ *and* $C1_{out}$ are the input and output states of a sub-task assigned to a dedicated cobot to shift the workpiece to the working zone.
- C_2 $C2_{in}$ *and* $C2_{out}$ is the action to paint the workpiece. This task is allocated to the second cobot in the system.

The painting process is abstracted using an extended MDP. The symbol $M_{painting}$ represents the system-level model as follows:

$$M_{painting} = (S, s_{init}, P, L, M_{c1}||M_{c2})$$

where $M_{painting}$ is the compositional system model, S is a five-state set of the system, $s_{init} \in S$ is the initial state S_1, $P : S \times S \times T \rightarrow [0, 1]$ is the transition matrix where $\sum_{s' \in S} P(s, s', t) = 1$ for all $s \in S, t \in T$ and $L : S \rightarrow 2^{AP}$ are function-labelling states with atomic propositions. $M_{c1}||M_{c2}$ are two cobot models. In this model, the cobot model is considered as a special state machine, in which the S_{in} and S_{out} of cobot model are the interfaces to integrate with the other four normal machine states S_1, S_2, S_3, S_4 and the *Failure* state S_5.

The initial transition matrix is defined below where the probability of Cobot I and II completing the task are $c1_{comp}$ and $c2_{comp}$ respectively. We use notation "/" to indicate nondeterministic choices in the state transition (5), shown at the bottom of the page.

Similar to the cobot child model, the initial transition matrix of the system model is defined according to the system specification and updated at run time based on the sensor confidence score.

D. EVALUATION OF SYSTEM FAILURE

Experiments were conducted to verify the safety and reliability of the painting process at run time using real sensor readings and working conditions. The probability of system failure is expressed by PCTL as below,

$$P_{failure} =? \, [\, F^{\leq 20} \, (S_5)]$$

where $P_{failure}$ is the probability of eventual system failure state, S_5, in the next 20 days, i.e., the service cycle of the cobot.

V. IMPLEMENTATION AND RESULTS

Two cobots were used to evaluate and validate the proposed compositional model verification framework. One was a UR10e [40] that worked as workpiece moving cobot C_1. The other was a Franka Emika [41], which was used for workpiece

$$P_{painting} = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c|ccccccc|c} & S_1 & S_2 & S_3 & C_1 & S_4 & C_2 & S_5 & \\ \hline & 0 & 1 & 0 & 0 & 0 & 0 & 0 & S_1 \\ & 0 & 0 & 1 & 0 & 0 & 0 & 0 & S_2 \\ & 0 & 0 & 1/0 & 0/1 & 0 & 0 & 0 & S_3 \\ & 0 & 0 & 0 & 0 & 0 & c1_{comp} & 1 - c1_{comp} & C_1 \\ & 0/0.9875 & 0 & 0 & 0 & 0 & 0.9875/0 & 0.0125 & S_4 \\ & 0 & 0 & 0 & 0 & c2_{comp} & 0 & 1 - c2_{comp} & C_2 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 1 & S_5 \end{array} \qquad (5)$$

painting cobot C_2. The sensor readings and cobot states were retrieved through Modbus protocol for the UR10e and a low level C++ interface, *libfranka*, for the Franka. The child model and painting system model were evaluated using a probabilistic model checker tool, PRISM [42].

In order to cover common and corner scenarios, six test cases were developed to evaluate the proposed approach.

- $TC1$ The common industry method that assumes the sensor's readings are 100% trusted.
- $TC2$ Using the proposed method that utilises run-time *sensor confidence score* to predict probability of failure of the painting system dynamically.
- $TC3$ Increasing the reading value by 10% over time to simulate drift event over time to validate the ability of the proposed framework to predict failure.
- $TC4$ Simulating another drift scenario that decreases the reading value by 10% over time to evaluate the algorithm.
- $TC5$ Manual intervention to disrupt the cobot movement and capture the sensor readings to evaluate the impact at system-level.
- $TC6$ Using a development firmware that returns only unsigned integer, which resulted in incorrect decoding of sensor reading of type signed integers. In most of the cases, this firmware worked well, however, when the reading exceeded 32,768, it caused overflow error and the control system made wrong decision according to the wrong readings.

We ran test case $TC1$, $TC5$ and $TC6$ once, while $TC2$, $TC3$, $TC4$ were executed in nine cycles of the painting process. Each cycle took about three minutes to complete the painting process.

At the beginning, we captured a dataset of machine states and sensor readings under the operator supervision to ensure the painting system works as expected. Subsequently, this dataset was processed to derive the *sensor normal behaviour* profile of the cobots to validate the test cases.

As the first test case *(TC1)* assumed that the sensor readings are 100% trusted, we obtained a system failure probability of 0.0125 at all times, which is not realistic as it is so close to perfect operation with zero failure. In *(TC2)*, we monitored the run-time sensor readings and then computed the *sensor confidence score* of 0.88 – 0.90, thus resulting in a system failure probability of 0.09 – 0.11. This effectively reflects the real situation of the painting system that it was not always working in the perfect condition. As for *(TC3)* and *(TC4)*, we simulated the drift events of ±10% of the real data that was captured in *(TC2)*, it is observed that the *sensor confidence score* dropped about 10% to 0.80, thus causing an increase in the failure probability to 0.15 and 0.18. Fig 6 summarises the comparison of test results of *(TC2), (TC3)* and *(TC4)*, comparing the real sensor values and the drift simulation results to evaluate their impacts on the failure probability with respect to the sensor confidence score. The horizontal axis indicates the cycle of the test cases and the vertical axis represents the sensor confidence score and the failure probability.

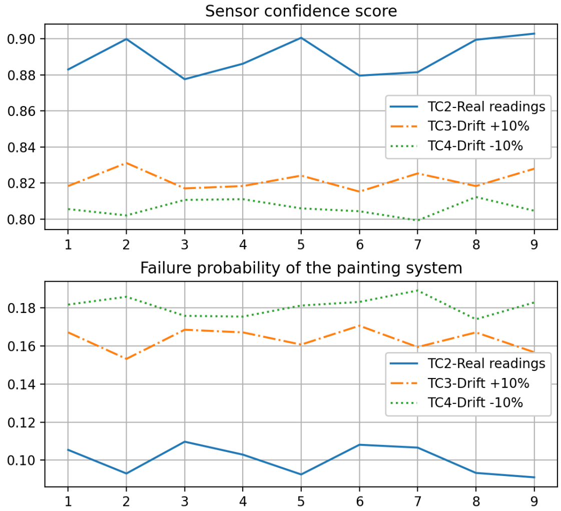

FIGURE 6. Comparison of real sensor readings and drift simulation.

TABLE 1. Sensor confidence score and failure probability.

Test Case	Sensor Confidence Score	Failure Probability
TC1	1.000	0.0125
TC5	0.6735	0.3635
TC6	0.3814	0.7905

Test case *(TC5)* was executed by blocking the cobot's action under safe guidance. In this case, we observed that the resulting *sensor confidence score* dropped to 0.6735 which is about 25% lower from the normal situation. With this, the 20-days failure probability had thus increased to 0.3635. Lastly, a defective firmware was used to evaluate the impact on the system failure probability, as such an error could result in the retrieval of incorrect sensor readings. Due to the incorrect decoding of signed integers, the *sensor confidence score* dropped to its lowest, 0.3814 and resulted in a high failure probability of 0.7905, as shown in Table 1. The full results are listed in Appendix I, Table 2.

VI. DISCUSSION

We have constructed a novel sensor network verification framework for compositional sensor network-based cyber-physical systems using two cobots as an evaluation use case. While the proposed framework can be used to predict potential system failure probability dynamically, model the sensor network's behaviour, and quantify the trustworthiness of a sensor network at the operation phase, we would also like to highlight several observations and insightful findings while evaluating the painting system use case. We provide further analysis and discussion of the experiment results and findings below:

1) The system failure probability is in inverse proportion to the sensor's *confidence score*. When the sensor's confidence score is low, this implies that there exhibits

anomalies and inaccuracy in the sensor readings obtained, and hence the readings are less trustworthy. Consequently, this has resulted in a higher probability of system failure than normal. In the case of the painting system, a linear function was applied to compute the sensor-network confidence score and then update the transition probability matrix. The choice of function very much depends on the logical relationship between the sensors and the system. This relationship can be configured and customised accordingly. For instance, machine learning techniques can be applied if the necessary dataset is available to determine the confidence score.

2) As the sensor is dynamic in nature, a transient fault in the sensor leading to the system failure will typically recover automatically. We observed that during one of the painting operations, the sensor confidence score unexpectedly dropped to 0.6808, thus increasing the potential failure probability to 0.3524. Nonetheless, the painting process was still completed as usual without any warning. However, we noticed that in the following operation cycle, the cobot indeed stopped working and reported a collision failure. This shows that our verification framework is able to predict the failure in a timely manner, thus warning the operators to pay more attention to such an event at run time in the future. As the failure could be due to the transient fault in the sensor, once the cobot was restarted, the painting system resumed to work smoothly.
3) In Industry 4.0, the manufacturing process should be flexible and formed by multiple components with minimal cost. With the proposed approach, the system-level verification reuses verified child models as much as possible to reduce the cost of creating the system model. Also, the knowledge of each independent child model can be explicitly represented and reused.

VII. CONCLUSION

The research we have conducted so far proposes a run-time compositional verification framework that combines data- and model-driven approaches. The framework takes into account sensor trustworthiness of each child model and the higher level system model can be formed by multiple child models. Hence, it enables the verification of the system reliability during the operation phase. We described how to dynamically update a probabilistic model with sensor network models that explicitly reflect sensor uncertainty. The methodology forms a unified run-time model that presents more accurate prediction results of impending system failures, even while the system is running. Furthermore, possible future work includes the following four directions:

1) The rules for deriving sensor network confidence scores are manually defined according to expertise on the specific system. This should be generalised so that the proposed approach can be applied to a wider range of systems.
2) A context-aware adaptive algorithm is needed to reflect more general scenarios, e.g., the cobot moves a heavy workpiece, diverse speed control.
3) This experiment focuses on sequential processes only. A hierarchical or more complex probabilistic model is required to model more sophisticated cyber physical systems. For instance, to verify multiple AGVs that carry workpieces to serve various processes concurrently.
4) ROS has become a popular real-time processing framework for CPSs over the last few years. However, ROS is not widely used in commercial applications. The lack of an efficient verification framework to verify the behaviour of such ROS-based systems could have hampered the adoption of ROS in the industry. The proposed verification framework could be extended to these ROS-based CPSs to guarantee performance, safety, and quality during operation.

The evaluation results highlighted the efficiency of explicitly modelling sensor trustworthiness, especially because automatically-collected sensor data might drive all consequential decisions in the domain of sensor network-based systems. Moreover, it helps the system operators to allocate and provision resources efficiently and in a more timely manner.

APPENDIX I. FULL EXPERIMENT RESULT

The experiment results of six scenarios.

TABLE 2. Sensor confidence score and failure probability.

Test Case	Sensor Confidence Score	Failure Probability
$TC1$	1.000	0.0125
$TC2$ dataset#1	0.883	0.1054
$TC2$ dataset#2	0.8999	0.093
$TC2$ dataset#3	0.8776	0.1097
$TC2$ dataset#4	0.8862	0.1029
$TC2$ dataset#5	0.9006	0.0925
$TC2$ dataset#6	0.8796	0.1081
$TC2$ dataset#7	0.8815	0.1066
$TC2$ dataset#8	0.8995	0.0933
$TC2$ dataset#9	0.9029	0.091
$TC3$ dataset#1	0.8183	0.1671
$TC3$ dataset#2	0.8311	0.1532
$TC3$ dataset#3	0.817	0.1685
$TC3$ dataset#4	0.8183	0.1671
$TC3$ dataset#5	0.8241	0.1607
$TC3$ dataset#6	0.8152	0.1706
$TC3$ dataset#7	0.8253	0.1594
$TC3$ dataset#8	0.8183	0.1671
$TC3$ dataset#9	0.8279	0.1567
$TC4$ dataset#1	0.8055	0.1817
$TC4$ dataset#2	0.802	0.1858
$TC4$ dataset#3	0.8106	0.1758
$TC4$ dataset#4	0.811	0.1754
$TC4$ dataset#5	0.8059	0.1812
$TC4$ dataset#6	0.8043	0.1831
$TC4$ dataset#7	0.7992	0.1891
$TC4$ dataset#8	0.8122	0.174
$TC4$ dataset#9	0.8046	0.1828
$TC5$	0.6735	0.3635
$TC6$	0.3814	0.7905

REFERENCES

[1] P. A. Lasota and J. A. Shah, "Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration," *Hum. Factors, J. Hum. Factors Ergonom. Soc.*, vol. 57, no. 1, pp. 21–33, Feb. 2015. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0018720814565188

[2] F. Gil-Vilda, A. Sune, J. A. Yagüe-Fabra, C. Crespo, and H. Serrano, "Integration of a collaborative robot in a U-shaped production line: A real case study," *Proc. Manuf.*, vol. 13, pp. 109–115, Jan. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2351978917306479

[3] M. Gleirscher, N. Johnson, P. Karachristou, R. Calinescu, J. Law, and J. Clark, "Challenges in the safety-security co-assurance of collaborative industrial robots," 2020, *arXiv:2007.11099*.

[4] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8401919/

[5] F. Tao, Q. Qi, L. Wang, and A. Y. C. Nee, "Digital twins and cyber–physical systems toward smart manufacturing and Industry 4.0: correlation and comparison," *Engineering*, vol. 5, no. 4, pp. 653–661, Aug. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S209580991830612X

[6] A. Felsberger and G. Reiner, "Sustainable Industry 4.0 in production and operations management: A systematic literature review," *Sustainability*, vol. 12, no. 19, p. 7982, 2020.

[7] X. Xin, S. L. Keoh, M. Sevegnani, and M. Saerbeck, "Dynamic probabilistic model checking for sensor validation in Industry 4.0 applications," in *Proc. IEEE Int. Conf. Smart Internet Things (SmartIoT)*, Beijing, China, Aug. 2020, pp. 43–50. [Online]. Available: https://ieeexplore.ieee.org/document/9191985/

[8] M. Calder and M. Sevegnani, "Stochastic model checking for predicting component failures and service availability," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 1, pp. 174–187, Jan. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/7812626/

[9] A. Filieri, C. Ghezzi, and G. Tamburrelli, "Run-time efficient probabilistic model checking," in *Proc. IEEE 33rd Int. Conf. Softw. Eng.*, May 2011, pp. 341–350. [Online]. Available: https://dl.acm.org/doi/10.1145/1985793.1985840

[10] N. Li, S. Adepu, E. Kang, and D. Garlan, "Explanations for human-on-the-loop: A probabilistic model checking approach," in *Proc. IEEE/ACM 15th Int. Symp. Softw. Eng. Adapt. Self-Manag. Syst.*, Jun. 2020, pp. 181–187.

[11] R. Bellman, "A Markovian decision process," *Indiana Univ. Math. J.*, vol. 6, no. 4, pp. 679–684, Apr. 1957.

[12] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli, "Model evolution by run-time parameter adaptation," in *Proc. IEEE 31st Int. Conf. Softw. Eng.*, Vancouver, BC, Canada, Jun. 2009, pp. 111–121. [Online]. Available: http://ieeexplore.ieee.org/document/5070513/

[13] H. L. S. Younes and R. G. Simmons, "Probabilistic verification of discrete event systems using acceptance sampling," in *Computer Aided Verification*, vol. 2404, G. Goos, J. Hartmanis, J. van Leeuwen, E. Brinksma, and K. G. Larsen, Eds. Berlin, Germany: Springer, 2002, pp. 223–235. [Online]. Available: http://link.springer.com/10.1007/3-540-45657-0_17

[14] L. Lestingi, M. Askarpour, M. M. Bersani, and M. Rossi, "A model-driven approach for the formal analysis of human-robot interaction scenarios," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Toronto, ON, Canada, Oct. 2020, pp. 1907–1914.

[15] E. M. Clarke and P. Zuliani, "Statistical model checking for cyber-physical systems," in *Automated Technology for Verification and Analysis*, vol. 6996, T. Bultan and P.-A. Hsiung, Eds. Berlin, Germany: Springer, 2011, pp. 1–12.

[16] H. L. S. Younes, M. Kwiatkowska, G. Norman, and D. Parker, "Numerical vs. statistical probabilistic model checking," *Int. J. Softw. Tools Technol. Transf.*, vol. 8, no. 3, pp. 216–228, Jun. 2006. [Online]. Available: http://link.springer.com/10.1007/s10009-005-0187-8

[17] M. Zarei, Y. Wang, and M. Pajic, "Statistical verification of learning-based cyber-physical systems," in *Proc. 23rd Int. Conf. Hybrid Syst., Comput. Control*. Sydney, NSW, Australia: ACM, Apr. 2020, pp. 1–7. [Online]. Available: https://dl.acm.org/doi/10.1145/3365365.3382209

[18] S. Mitsch and A. Platzer, "ModelPlex: Verified runtime validation of verified cyber-physical system models," *Formal Methods Syst. Design*, vol. 49, nos. 1–2, pp. 33–74, Oct. 2016. [Online]. Available: http://link.springer.com/10.1007/s10703-016-0241-z

[19] A. Platzer, "Differential dynamic logics: Automated theorem proving for hybrid systems," *Künstliche Intelligenz*, vol. 24, no. 1, pp. 75–77, Apr. 2010. [Online]. Available: http://link.springer.com/10.1007/s13218-010-0014-6

[20] R. Bohrer, Y. K. Tan, S. Mitsch, A. Sogokon, and A. Platzer, "A formal safety net for waypoint-following in ground robots," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2910–2917, Jul. 2019.

[21] S. Mitsch, K. Ghorbal, D. Vogelbacher, and A. Platzer, "Formal verification of obstacle avoidance and navigation of ground robots," *Int. J. Robot. Res.*, vol. 36, no. 12, pp. 1312–1340, Oct. 2017. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364917733549

[22] R. Bohrer, A. Luo, X. A. Chuang, and A. Platzer, "CoasterX: A case study in component-driven hybrid systems proof automation," *IFAC-PapersOnLine*, vol. 51, no. 16, pp. 55–60, 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2405896318311236

[23] N. Fulton, S. Mitsch, J.-D. Quesel, M. Völp, and A. Platzer, "KeYmaera X: An axiomatic tactical theorem prover for hybrid systems," in *Automated Deduction—CADE*, vol. 9195, A. P. Felty and A. Middeldorp, Eds. Cham, Switzerland: Springer, 2015, pp. 527–538. [Online]. Available: http://link.springer.com/10.1007/978-3-319-21401-6_36

[24] J.-D. Quesel, S. Mitsch, S. Loos, N. Aréchiga, and A. Platzer, "How to model and prove hybrid systems with KeYmaera: A tutorial on safety," *Int. J. Softw. Tools Technol. Transf.*, vol. 18, no. 1, pp. 67–91, Feb. 2016. [Online]. Available: https://link.springer.com/10.1007/s10009-015-0367-0

[25] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, Eds., *Handbook of Model Checking*. Cham, Switzerland: Springer, 2018. [Online]. Available: http://link.springer.com/10.1007/978-3-319-10575-8

[26] N. Kamide and R. Yano, "Logics and translations for hierarchical model checking," *Proc. Comput. Sci.*, vol. 112, pp. 31–40, Dec. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877050917313534

[27] A. Desai, T. Dreossi, and S. A. Seshia, "Combining model checking and runtime verification for safe robotics," in *Runtime Verification*, vol. 10548, S. Lahiri and G. Reger, Eds. Cham, Switzerland: Springer, 2017, pp. 172–189. [Online]. Available: http://link.springer.com/10.1007/978-3-319-67531-2_11

[28] J. Huang, C. Erdogan, Y. Zhang, B. Moore, Q. Luo, A. Sundaresan, and G. Rosu, "ROSRV: Runtime verification for robots," in *Runtime Verification*, vol. 8734, B. Bonakdarpour and S. A. Smolka, Eds. Cham, Switzerland: Springer, 2014, pp. 247–254. [Online]. Available: http://link.springer.com/10.1007/978-3-319-11164-3_20

[29] A. Ferrando, R. C. Cardoso, M. Fisher, D. Ancona, L. Franceschini, and V. Mascardi, "ROSMonitoring: A runtime verification framework for ROS," in *Towards Autonomous Robotic Systems*, vol. 12228, A. Mohammad, X. Dong, and M. Russo, Eds. Cham, Switzerland: Springer, 2020, pp. 387–399. [Online]. Available: http://link.springer.com/10.1007/978-3-030-63486-5_40

[30] P. C. Attie and N. A. Lynch, "Dynamic input/output automata: A formal and compositional model for dynamic systems," *Inf. Comput.*, vol. 249, pp. 28–75, Aug. 2016, doi: 10.1016/j.ic.2016.03.008.

[31] P. Civit and M. Potop-Butucaru, "Probabilistic dynamic input output automata," Leibniz Int. Proc. Inform. (LIPIcs), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2022, pp. 15:1–15:18, vol. 246. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2022/17206

[32] K. Ni, N. Ramanathan, M. N. H. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava, "Sensor network data fault types," *ACM Trans. Sensor Netw.*, vol. 5, no. 3, pp. 1–29, May 2009. [Online]. Available: https://dl.acm.org/doi/10.1145/1525856.1525863

[33] A. B. Sharma, L. Golubchik, and R. Govindan, "Sensor faults: Detection methods and prevalence in real-world datasets," *ACM Trans. Sensor Netw.*, vol. 6, no. 3, pp. 1–39, Jun. 2010. [Online]. Available: https://dl.acm.org/doi/10.1145/1754414.1754419

[34] D. Park, S. Kim, Y. An, and J.-Y. Jung, "LiReD: A light-weight real-time fault detection system for edge computing using LSTM recurrent neural networks," *Sensors*, vol. 18, no. 7, p. 2110, Jun. 2018. [Online]. Available: http://www.mdpi.com/1424-8220/18/7/2110

[35] D. Li, Y. Wang, J. Wang, C. Wang, and Y. Duan, "Recent advances in sensor fault diagnosis: A review," *Sens. Actuators A, Phys.*, vol. 309, Jul. 2020, Art. no. 111990. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924424719308635

[36] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal Aspects Comput.*, vol. 6, no. 5, pp. 512–535, Sep. 2004.

[37] *Robots and Robotic Devices—Collaborative Robots*, Standard ISO/TS 15066:2016(en), 2016. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:ts:15066:ed-1:v1:en

[38] M. Gurgul, *Industrial Robots and Cobots: Everything You Need to Know About Your Future Co-Worker*. USA: Michal Gurgul, 2019.

[39] J. Buckner, B. L. Chin, and J. Henri, ''Prometrix RS35e gauge study in five two-level factors and one three-level factor,'' in *Statistical Case Studies for Industrial Process Improvement*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1997, pp. 9–17. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898719765.ch2

[40] *UR10e Collaborative Industrial Robot*. Accessed: Nov. 30, 2022. [Online]. Available: https://www.universal-robots.com/products/ur10-robot/

[41] *Franka Emika*. Accessed: Nov. 30, 2022. [Online]. Available: https://www.franka.de/production/

[42] M. Kwiatkowska, G. Norman, and D. Parker, ''PRISM 4.0: Verification of probabilistic real-time systems,'' in *Computer Aided Verification*, vol. 6806, G. Gopalakrishnan and S. Qadeer, Eds. Berlin, Germany: Springer, 2011, pp. 585–591.

XIN XIN is currently pursuing the Ph.D. degree with the School of Computing Science, University of Glasgow. Before he started the Ph.D., he was a Research Engineer at the Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR) Singapore, focus on distributed computing and social cognition computing. He also works at TÜV SÜD Digital Service, Singapore, as a Principal Engineer and leads the software development of a run-time verification engine, which takes into account sensor run-time trustworthiness to verify sensor network-based systems' behavior. His research interests include sensor trustworthiness quantification and impacts analysis for safety-critical systems.

SYE LOONG KEOH (Member, IEEE) received the Ph.D. degree in computing from the Imperial College London.

He is currently an Associate Professor at the School of Computing Science, University of Glasgow (UofG), Singapore. He is also the Programme Director of the SIT-UofG Joint Degree in computing science and the Director of Research Programmes at UofG. Prior to joining Glasgow, he was a Senior Scientist at Philips Research Eindhoven, The Netherlands (Philips Lighting is currently known as Signify). He leads the cyber-security research activities at UofG Singapore, where he has designed several lightweight authentication protocols and key management schemes for the IoT, building management, and industrial control systems. He is currently researching on new techniques for securing end-to-end communication and ensuring data provenance in the IoT environment. While working at Philips Research, he was responsible for standardizing Marlin Digital Rights Management (DRM) technology for content protection and lightweight security protocols for the Philips's IoT-based lighting systems. His research interests include cyber security for the Internet of Things (IoT), lightweight security systems for cyber-physical systems, and policy-based security management for pervasive and distributed systems.

MICHELE SEVEGNANI received the M.Sc. degree in bioinformatics from The University of Edinburgh, in 2008, and the Ph.D. degree in computing science from the University of Glasgow, Glasgow, U.K., in 2012. He is currently a Senior Lecturer of computing science at the University of Glasgow. He is also a Principal Investigator of research projects on modeling perspectives in autonomous vehicles and resilience assurance in agritech systems. He was a Visiting Researcher at the University of California at Berkeley, on modeling autonomous swarm systems. His research interests include online models, stochastic models for predicting failures and availability, sensor validation in the IoT, and reasoning for autonomous agents.

MARTIN SAERBECK received the degree in computer science and the Ph.D. degree in industrial design. He is currently a CTO of Digital Service at TÜV SÜD, where he leads strategic research and development initiatives of novel digital testing solutions in the domains of AI, robotics, and the IoT technology. He has over 15 years of experience in developing technical solutions for both industry and academia. After starting his career at Philips Research, he established a new research group at A*STAR IHPC and delivered innovation projects in the sectors of aerospace, manufacturing, and retail. He has a passion for applied research, promoting the translation of academic results to make today's connected smart systems safe, secure, and reliable.

TECK PING KHOO was born in Singapore, in 1980. He received the Bachelor of Engineering degree in electrical from the National University of Singapore (NUS), in 2005, the Master of Science degree in digital media technology from Nanyang Technological University (NTU), in 2012, and the Ph.D. degree in software engineering from the Singapore University of Technology and Design (SUTD), in 2021. From 2005 to 2013, he was a Network Protocol Developer at DSO National Laboratories, Singapore. From 2013 to 2016, he lectured a Python programming at Republic Polytechnic, Singapore. Since 2016, he has been a Software Quality Engineer at TÜV SÜD Asia–Pacific, Singapore. He is the author of two research papers, which are about active and passive learning of software. His research interests include model based testing of cyber physical systems, software quality assurance (SQA), robotics, and the IoT.