Dynamic Probabilistic Model Checking for Sensor Validation in Industry 4.0 Applications

Xin Xin*†, Sye Loong Keoh*, Michele Sevegnani*, Martin Saerbeck†
*School of Computing Science, University of Glasgow, Glasgow, United Kingdom
Email: {SyeLoong.Keoh, Michele.Sevegnani}@glasgow.ac.uk
†Digital Service, Centre of Excellence, TÜV SÜD Asia Pacific, Singapore
Email: {Xin.Xin, Martin.Saerbeck}@tuv-sud.sg

***Abstract*—Industry 4.0 adopts Internet of Things (IoT) and service-oriented architectures to integrate Cyber-Physical Systems and Enterprise Planning into manufacturing operations. This kind of integration consists of a combination of connected sensors and run-time control algorithms. Consequential control decisions are driven by sensor-generated data. Hence, the trustworthiness of the sensor network readings is increasingly crucial to guarantee the performance and the quality of a manufacturing task. However, existing methodologies to test such systems often do not scale to the complexity and dynamic nature of today's sensor networks. This paper proposes a novel run-time verification framework combining sensor-level fault detection and system-level probabilistic model checking. This framework can rigorously quantify the trustworthiness of sensor readings, hence enabling formal reasoning for system failure prediction. We evaluated our approach on an industrial turn-mill machine equipped with a sensor network to monitor its main components continuously. The results indicate that the proposed verification framework involving the quantified sensor's trustworthiness enhances the accuracy of the system failure prediction.**

***Index Terms*—Probabilistic Model Checking, Sensor Confidence, Sensor Trustworthiness, IIoT, Industry 4.0**

I. Introduction

Recent advancements in embedded sensor systems and Artificial Intelligence [1]–[3] have achieved a significant improvement of modern smart manufacturing systems in terms of intelligent controls, predictive analytics and system automation. The ability to collect shop-floor data at run-time has enabled monitoring of each individual machines and deriving insights that can subsequently be translated into self-control functions, hence automating the operations of the manufacturing plant. The data collected from the machines at run-time (including machine status and work piece properties) are then used to continuously update machine configuration parameters. Thus, such production systems are forming a closed feedback-control loop with sensor data becoming the key driver to maintain smooth operation of the manufacturing plant.

There are two general challenges that need to be addressed in order to ensure smooth machine operation in a smart manufacturing plant. Firstly, there must be an efficient approach to accurately model and compare machine status with its expected behaviour over time. Timely identification of deviations is an essential prerequisite to significantly reduce machine down time and to manage the resources for repairing faulty components in a cost-effective manner. Secondly, the accurate prediction of a machine behaviour depends on an accurate data acquisition from the sensors instrumenting the machine. This implies that the *trustworthiness* of the sensor readings is crucial. All maintenance and control decisions are only as good as the sensor data that they are based on. However, current predictive maintenance algorithms tend to assume that sensor readings are accurate and reliable, which is often not the case.

In real-world deployments, it is inevitable that inaccurate readings might occur, for example due to calibration issues, sensor wear and tear, or even malicious tampering, leading to deviation from the actual values. The use of these inaccurate and untrustworthy readings in production lines can significantly degrade product quality as well as affect the performance of the manufacturing system as a whole. Currently deployed methods to monitor machines in automation tend to ignore the dynamic nature of the sensor networks as well as the possibility of sensor failures.

While monitoring of machines is facilitated by sensors, a different approach is required to automate the monitoring sensor networks themselves. After all, manufacturers are not going to stick sensors on sensors for economic and operational reasons. Traditionally, manufacturers have employed a time based, often manual maintenance regime to regularly test and calibrate sensors with special tools. In the absence of such tools and considering the number of sensors driving Industry 4.0 applications, a new sensor network monitoring and testing framework is needed.

This paper proposes a novel run-time verification framework to provide quantified trustworthiness for sensor-network-based systems, by combining data-driven [4] and model-driven [5] techniques. Model-driven techniques are typically used to provide an abstraction of a complex system, thus enabling the users to visualise, predict, optimise, regulate and control the system effectively [6]. All models are simplifications derived from a deep understanding of the system or a scientifically established relation between its components and the environment. The output of such models is only as good as the assumptions and data that they operate on. As the data and

M. Sevegnani is supported by EPSRC grant S4: Science of Sensor Systems Software (EP/N007565/1) and PETRAS SRF grant MAGIC (EP/S035362/1).
X. Xin is partially funded by Singapore Economic Development Board (EDB) through the Industrial Postgraduate Programme (IPP) Grant.

assumptions change over time, the models should also be updated and reviewed accordingly to reflect the actual behaviour of the system. By integrating data-driven approaches with model-driven approaches, as well as an established software testing and cybersecurity principles that ensure authenticity and trustworthiness of datasets (both historical and run-time), it is now feasible to refine and accurately improve the models over time. In this paper, a CNC (Computer Numerical Control) turn-mill machine is used as a test-bed to evaluate this new approach.

The contributions of this paper are summarised as follows:

- We propose a novel run-time verification framework is designed and implemented to provide quantified trustworthiness for sensor-network-based systems.
- This framework combines both sensor-level data-driven models and a system-level probabilistic model. Sensor-level models are used to quantify the *trustworthiness* of each sensor for the computation of transition probability. The probabilistic model is a system abstraction to reflect the working machine status in run-time and predict system failure in a certain period.
- This verification framework is validated using a real-life CNC turn-mill machine, involving forty-three sensors and five machine states.

This paper is organised as follows: Section II provides an overview of background and related work. Section III presents a dynamic probabilistic model to continuously verify a sensor network-based system. Section IV introduces an implementation to predict the failure probability of a sensor network enabled CNC turn-mill machine. The experiment and results are presented in Section V. Finally, we conclude the paper with future work in Section VI.

II. Related Work

A. Types of Sensor Data Fault

Sensor faults are very common in sensor network-based systems, thus producing unreliable data that does not reflect the true state of the machine or environment. *Ni et al.* [7] defined a systematically characterized taxonomy of sensor data faults into three categories, namely *environment features*, *system features or specifications*, and *data features*. Such classification helps to identify and develop fault detection methods.

- *Environment Features* capture context and operational conditions of a sensor, including where the sensor is placed, temperature and humidity of the environment. It also includes measured modality of the sensor and its boundary conditions. These features are crucial in determining the expected behaviour which can then be used to determine sensor or system faults.
- *System Features or Specifications* include hardware components and calibration features. Hardware components describe the abilities of a sensor. Calibration describes the uncertainty of the mapping from input to output. For example, clipping may be a hardware or software limitation exhibited by a sensor when it maxes out due to operating conditions exceeding the limits or calibration conditions.
- *Data Features* are usually statistical in nature and calculated in either the spatial or temporal domain. For example, sensors are expected to retain similar characteristics as other co-located sensors, and operate reliably for a period of time.

The *data features* is of particular interest as it includes two groups of typical sensor failures. One is observable from a data-centric point of view, where outliers, spikes and "stuck-at" faults can be identified statistically. The other group of sensor faults requires taking a system-centric view, this includes, among others, calibration faults, connection or hardware failures, low battery, out of range failures and clipping.

B. Detection of Sensor Faults

Sharma et al. [4] summarised four methods of sensor faults detection, which are rule-based, time series analysis-based, learning-based and estimation methods.

- The *Rule-based* approach is a common method that uses domain knowledge of sensor behaviour to develop heuristic rules to ensure that the sensor readings are in accordance with the modelled expectation.
- The *Time series analysis-based* approach uses temporal correlation in measurements to build a normal behaviour sensor model. Subsequently, data collected from the sensors in operation is compared against this normal behaviour model, often using time series forecasting to determine whether they are faulty.
- The *Learning-based* approach builds a sensor module to analyse the normal and faulty behaviours from historical data using statistical modelling or machine learning techniques. It often attempts to perform a root cause analysis. Based on the resulting inference models, sensor faults can be detected and the reason of fault can be inferred as well.
- The *Estimation* approach exploits spatial and temporal correlations in measurements from different sensors to generate the normal sensor behaviour.

Although this can help providing a guideline to detect sensor faults, it is still a challenge to quantify the impact of faulty sensors on the system as a whole.

Ramanathan et al. introduced an application, which applied sensor-level faults detection methods in an environmental monitoring sensor networks [8]. After forty-eight sensors were deployed in the field, they found that a significant amount of sensor readings was uninterpretable due to the prevalence of anomalous patterns. In order to identify the faults, a fault detection system was developed, which applied pre-configured rules to detect invalid data and identify faulty sensors. While it helps to verify the data integrity at sensor-level, it does not reflect the impact of the sensor faults at system level.

C. Probabilistic Model Checking

Kwiatkowska et al. [5] used the probabilistic model checker PRISM [9] to abstract an Internet-of-Things (IoT) system with a probabilistic model. The resulting model is used to evaluate the performance and reliability of a sensor-network-based system with the injection of one or more sensor failures.

Calder et al. [6] proposed a methodology for failure prediction of a critical communication system using a probability model. It defines three status categories for each component within the system, *working, reduced-redundancy and no-service*. The model predicts not only the probability of system failure based on each component's status, it also predicts future service availability. This methodology helps the operators to allocate resources optimally in the presence of component failures.

Sevegnani et al. [10] demonstrated a modelling and verification framework of a large-scaled sensor network system. This framework is able to capture not only the spatial, operational-aspects but also the temporal evolution, which is the dynamic behaviour of the sensor network. This approach is capable of handling online verification of large-scale sensor network-based systems.

We are extending these works by integrating run-time detection of sensor faults so that the probabilistic model can be updated dynamically, providing an accurate representation of the physical system's behaviour.

III. Dynamic Probabilistic Model Checking

We propose a novel approach to dynamically model a sensor network-based system using Discrete-Time Markov Chains (DTMCs) and the probabilistic model checker PRISM [11], with the ability to detect sensor faults at run-time.

A base-model is first defined by a domain expert to provide a system level abstraction and verification of the system behaviour. This means manually specifying the initial transition probability matrix of the DTMC. The model then evolves over time as the system continuously learns about the sensors behaviour through profiling and analysis of past sensor data. In order to ensure the *trustworthiness* of the data collected from the sensors, the sensor data is compared against its *normal behaviour* using time series analysis and estimation methods, so that any deviation from the norm can be detected in run-time. Once a sensor fault is detected, the transition matrix of the base-model is updated with a lower *confidence* in the trustworthiness of the sensor data. Essentially, this means that the model is updated continuously, taking into account run-time sensor data to derive the appropriate probability of state transitions. Hence, the methodology tracks both, the sensor readings and the expectation for comparison and verification. It enables the verification of the system reliability at run-time and at the same time predicts potential system failures.

Fig. 1 shows an architectural overview of the proposed dynamic probabilistic model checker, integrating both the model-driven and data-driven approaches. It is represented by two modules, namely *System Model Verification (SMV)* and *Sensor Fault Detection (SFD)*. The *System Under Test (SUT)* is the physical sensor network-based system, streaming run-time sensor readings to SMV and SFD. In addition, the sensor data and the system states are saved into a *historical data store* to facilitate the learning of normal sensor behaviour.

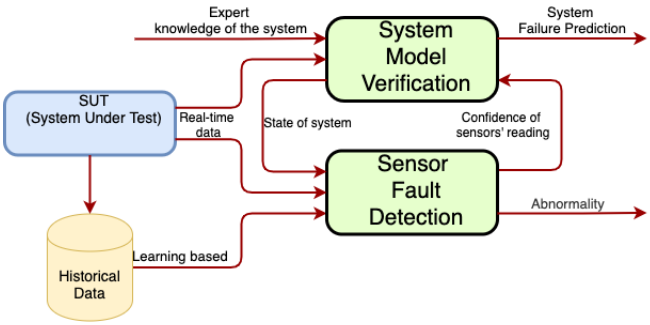

Fig. 1. Architecture of the dynamic probabilistic model checker.

A. Sensor Fault Detection (SFD)

SFD is based on a set of quality evaluation metrics that is built from historical sensor data. In combination with a rule engine, SFD is able to quantify the trustworthiness of the sensor data at run-time. As shown in Fig. 2, the SFD consists of two major modules, namely *Sensor Behaviour Analyser (SBA)*, and the *Rules Engine*.

SBA learns about the sensor behaviour from the historical data periodically i.e., on a daily basis. In addition, SBA analyses run-time data obtained from the sensor reading pipeline in order to determine data patterns. With this, a sensor profile termed as *sensor normal behaviour* can be derived. Each *sensor normal behaviour* comprises the following three elements:

- *Statistical characteristics* — calculated over a window of samples. Windowing is done over the temporal domain on an individual sensor basis. The *mean* and *standard deviation* are the basic statistical measures, often used to measure the reliability of a sensor. When the standard deviation is high (relative to expectation), it adds evidence to be classified as faulty data [12].
- *Estimation model* — uses data mining techniques, e.g., machine learning methods, to forecast data range and reading patterns. This process does not require knowledge from domain experts.
- *Drift trend* — is the direction at which the sensor reading is moving, usually away from its normal behaviour. Drift is typically a result of sensor wear and tear and calibration errors [7]. As the sensor's readings are in time series format, an ARIMA [13] model is used to calculate the trend component in order to determine whether there are consistent deviations (in increasing or decreasing order) in the sensor readings over time.

The second module, the *Rules Engine* determines a sensor's *confidence score* based on the run-time sensor readings obtained from the pipeline by evaluating the degree of deviation from the *sensor normal behaviour*. As the ground truth is not

TABLE I
SENSOR FAULTS AND DETECTION METHODS.

Fault Type	Description	Detection Method
Outlier	Outliers in data are one of the most common sensor faults. The sensor readings are out of range relative to expectation.	Rule-based
Spike	The readings are changing more than expected over a short period of time. It may or may not return to normal afterwards.	Rule-based
Stuck-at/Constant	The readings are kept the same or almost the same for a longer period of time than expected.	Rule-based
Intermittent	Deviations from normal readings appear and disappear several times, the frequency of this signature is generally random.	Rule-based
High Noise/Variance	Noise is common and expected in sensor data. When unusually high noise is caused by hardware failure or low batteries, the data may differ from expectations.	Rule-based
Bias	A constant offset from the ground truth. The governing equation shall be $Y_{reading} = X + b + variance$, where X is the ground truth and b is the constant offset.	Correlation-based
Drift	A time-varying offset from the ground truth of sensor's signal. The equation should be $Y_{reading} = X + f(t) + variance$, where X is the ground truth and f(t) is time-varying offset.	Time series analysis-based
Scale	A time-varying offset from the ground truth of sensor's signal. The equation should be $Y_{reading} = X \times f(t) + variance$, where X is the ground truth and f(t) is time-varying offset.	Time series analysis-based

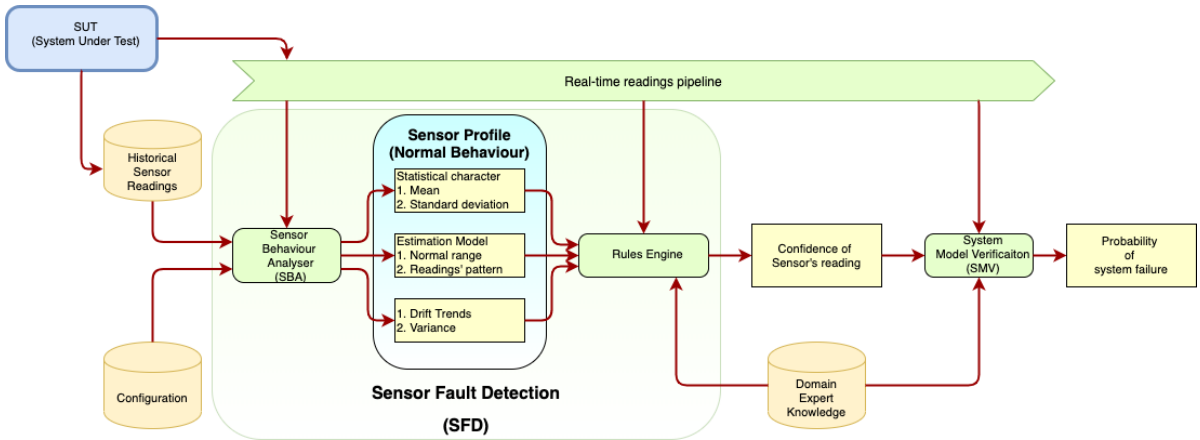

Fig. 2. Modules of the dynamic probabilistic model checker.

known, this deviation is usually termed as *fault*. Table I shows a list of fault types that are defined and can be detected by SFD. In order to detect faults, a set of rules are defined based on the domain expert knowledge, so that all incoming run-time readings are evaluated against these pre-configured rules and the *sensor normal behaviour* to derive the *confidence score*. A sensor's *confidence score* is thus calculated as follows:

$$Conf_{sensor} = \sum_{i=0}^{n} factor_i \times weight_i \tag{1}$$

where n is number of the rules defined, $factor_i$ is the output of each rule i and $weight_i$ is the weight of the rules.

Note that the *sensor normal behaviour* comprises the three elements that are used to compute the sensor's *confidence score*, which is a representation of the trustworthiness of the sensor readings. This *confidence score* is subsequently fed into the SMV (c.f. Section III-B) to dynamically update the transition matrix of the model.

B. System Model Verification (SMV)

SMV is a probabilistic model-based system abstraction to verify the working behaviour of SUT and to predict system failure in real time. It is a DTMC model defined as follows:

$$M_{system} = (S, s_{init}, P, L) \tag{2}$$

where M_{system} is the probabilistic model of SUT, S is a finite set of machine states of SUT, $s_{init} \in S$ is the initial state, $P : S \times S \to [0, 1]$ is the transition probability matrix where $\sum_{s' \in S} P(s, s') = 1$ for all $s \in S$ and $L : S \to 2^{AP}$ are function-labelling states with atomic propositions.

With this definition, the states and transitions of the model are fixed and do not change during run-time, only the probability of state transition in the model is dynamically updated according to the run-time sensor readings and its working conditions [7]. The initial probability of state transitions in the model must be defined based on the domain expert knowledge. Subsequently, the transition probability matrix P evolves over time according to the sensor's *confidence score* obtained from

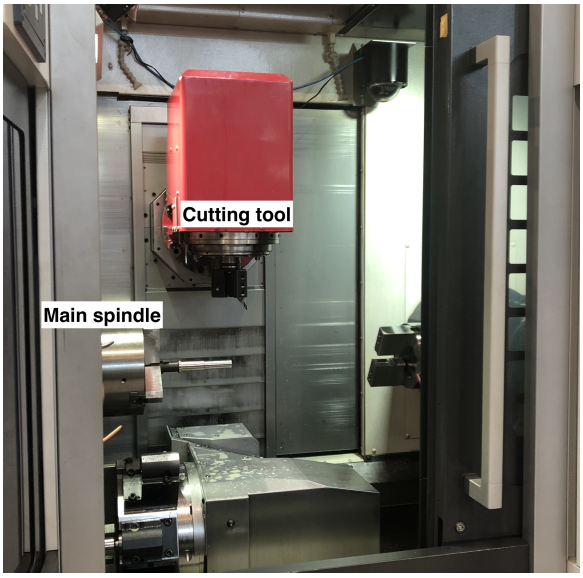

Fig. 3. The location of sensors in a CNC turn-mill machine.

SFD. This effectively models the behaviour of the overall system more accurately whenever faults are detected in the sensor readings, and thus leading to better prediction of system failure.

We will describe how the sensor confidence score is computed and how it updates the transition probability matrix in the next section.

IV. Implementation

We modelled a CNC (Computer Numerical Control) turn-mill machine using the proposed dynamic probabilistic model checker. Fig. 3 shows a CNC turn-mill machine NTX1000 [14], that is equipped with a sensor network to monitor its main spindle and cutting tool. The sensor network consists of three types of sensors, namely *current* sensors, *vibration* sensors and *temperature* sensors. These sensors are connected to a data acquisition system, the extracted sensor features are streamed to the server via OPC-UA [15] protocol.

We implemented the proposed dynamic probabilistic model checker for CNC turn-mill machine based on the data flow and the software modules presented in Fig. 2. In this section, we presents the system assumptions and the implementation details of SMV and SFD.

A. Sensor Fault Detection (SFD) Module

SFD was implemented using Python and the interfaces were developed following RESTful [16] architecture patterns. Historical data and configurations are provided through a web-based interface. SFD inspects historical readings to extract sensor normal behaviour on a daily basis. During run-time, the sensor readings are generated by SUT and published to *run-time readings pipeline*. With this, all processing modules can subscribe to the readings pipeline to obtain the data. SFD receives the run-time readings to compute a confidence score

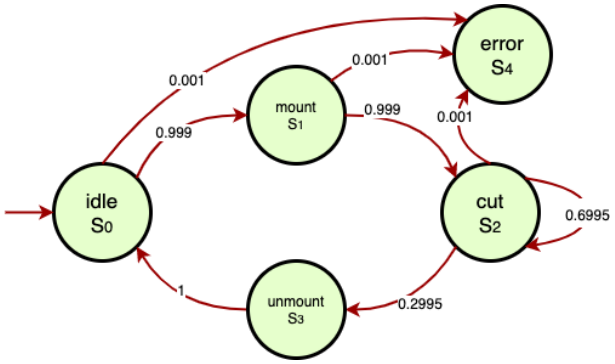

Fig. 4. The state transition model of CNC turn-mill machine.

for each sensor against its normal behaviour. Afterwards, this confidence score is fed to SMV to update the transition matrix.

B. System Model Verification (SMV) Module

The SMV was implemented using PRISM 4.5 with a Java wrapper. In order to interact with the SFD, a web server was set up as a container to run the SMV module and to exchange parameters.

A system-level probabilistic model was developed by domain experts based on the domain knowledge and operating experiences. We describe an illustrative case to evaluate the proposed approach. Fig. 4 shows the system model, which includes five states:

$$AP = \{idle, mount, cut, unmount, error\} \tag{3}$$

and the initial transition matrix is defined as:

$$P_{init} = \begin{bmatrix} 0 & 0.999 & 0 & 0 & 0.001 \\ 0 & 0 & 0.999 & 0 & 0.001 \\ 0 & 0 & 0.6995 & 0.2995 & 0.001 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4}$$

where the probability of each state transition in the matrix is defined based on expert knowledge and historical data. The initial probability of transition matrix, P_{init} assumes that the sensors are new and behave normally. The probability of system failure is expressed by the following PCTL [17] formula:

$$P_{system_{error}} =? \left[F^{\leq 60\times60\times24\times7} (S_4) \right] \tag{5}$$

where $P_{system_{error}}$ is the probability of system error in seven days. The error state S_4 is defined in the machine state in Fig. 4.

Ideally, the model should accurately reflect the system state. In practice, however, the sensor network-based system might be different. This is a crucial step that existing methods tend to ignore. As discussed, the deviation may be explained, among others, by wear and tear of the sensors or sensor readings drift. In order to make the system model dynamic (updating evaluation and expectation in unison), the probability of each

transition can be updated according to the run-time *sensor confidence score* of each sensor from the SFD. Since each machine state is monitored by multiple sensors, the probability of transition to an error state should be included in the overall sensors' confidence score. The failure rate of each state will be updated according to the following:

$$R_{failure} = \sum_{i=0}^{n} (1 - conf_i) \times w_i \times \lambda_i \tag{6}$$

where the transition to error state depends on the trustworthiness of n sensors, $conf_i$ is confidence score of sensor i obtained from SFD, w_i is the weight assigned to sensor i in the current state and λ_i is Mean-Time-To-Failure (MTTF) of sensor i.

Consequently, by knowing $R_{failure}$, the system's initial probability model, P_{init} can be dynamically updated whenever sensor faults are detected. We define $P_{dynamic}$ as a continuously updated probability transition matrix of the system as follows:

$$P_{dynamic} = \begin{bmatrix} 0 & 1 - R_{failure} & 0 & 0 & R_{failure} \\ 0 & 0 & 1 - R_{failure} & 0 & R_{failure} \\ 0 & 0 & 0.7 \times (1 - R_{failure}) & 0.3 \times (1 - R_{failure}) & R_{failure} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where the coefficients 0.7 and 0.3 are defined based on domain expert's experience for this specific turn-mill machine. Hence, domain expertise is not only used for the initial model, but also guides the update of expectations. With the run-time sensor confidence score, the system failure probability of the whole system should reflect the real system states more accurately.

C. Assumptions

The proposed dynamic probabilistic model for the turn-mill machine was implemented based on the assumptions defined below:

1) The initial transition probability matrix, P_{init} is first defined by experienced operators. After the system started its operation, the probability of transition, $P_{dynamic}$ is updated dynamically based on the *sensor confidence score*.
2) The system model of this implementation is based on two modules only, which are the main spindle and cutting tool.
3) Each module is monitored by three types of sensors: *current, vibration* and *temperature* sensor.
4) An experienced operator sets the weights of the three sensor contributions of the module as
   a) Current sensor: 0.3
   b) Vibration sensor: 0.5
   c) Temperature sensor: 0.2
5) Initial *sensor confidence score* was set as 0.99 for all sensors.
6) To simplify the complexity of model, all sensors' MTTF are assumed to be 100,000 hours.

V. Experimental Results

We observed three machine states, *MOUNT, CUT* and *UNMOUNT* to evaluate our implementation.

It is clear that the sensors exhibited different patterns at different machine states. Fig. 5 shows the readings of *current* sensor, *temperature* sensor and *vibration* sensor from the main spindle. During the transition from *MOUNT* to *CUT*, and then to *UNMOUNT* state, we observed that the sensors showed three different patterns. At the *MOUNT* stage, the temperature and vibration readings were relatively stable, while the current was almost zero. Subsequently, When the machine started the cut operation, we observed that the temperature reading increased consistently and the vibration amplitude was much greater. Once the cutting job had been completed, the machine moved to the *UNMOUNT* state, and it is observed that the vibration re-stabilised. The temperature reading remained elevated while the current returned to zero.

A. Sensor Fault Detection (SFD)

In this implementation, we segregated the sensor readings according to the machine states in order to determine the *sensor normal behaviour*. For each sensor's normal behaviour model in each machine state, the same algorithm was used to analyse the readings. Tables II show the snapshots of two machine states: *MOUNT* and *CUT*, where all three sensors' readings were analysed.

In SFD, there is a *Rules Engine* that is responsible for detecting sensor faults. Having defined the sensor's normal behaviour in Table II, we further configured the following rules in line with the domain of the case-study to evaluate the concept. The exact parameter values are of less importance and hence are chosen conservatively. It is expected that the performance can be improved by fine-tuning the values. To illustrate the methodology, the following example rules suffice:

1) If there is zero variation (standard deviation) in the sensor readings for more than 30 seconds, the sensor is deemed as having Stuck At fault.
2) According to the experiences, if more than 50% of the readings are missing, the sensor is deemed as having an Intermittent fault. Otherwise, the percentage ratio of received readings and the total number of expected sensor readings is returned.
3) Compute the distance between the sensor reading and the mid point of the estimated range in the sensor's normal behaviour model.
4) If drift trend is greater than 0.5, the sensor is deemed as having a Drift fault.

Table III shows a snapshot of a sensor confidence score during the CUT stage. Based on the confidence score, it is rather easy to identify the working condition of each sensor. However, it is quite difficult to determine how a sensor affects the machine's overall working status. With the proposed dynamic probabilistic model checking, the resulting confidence scores are used to update the transition matrix of the model.

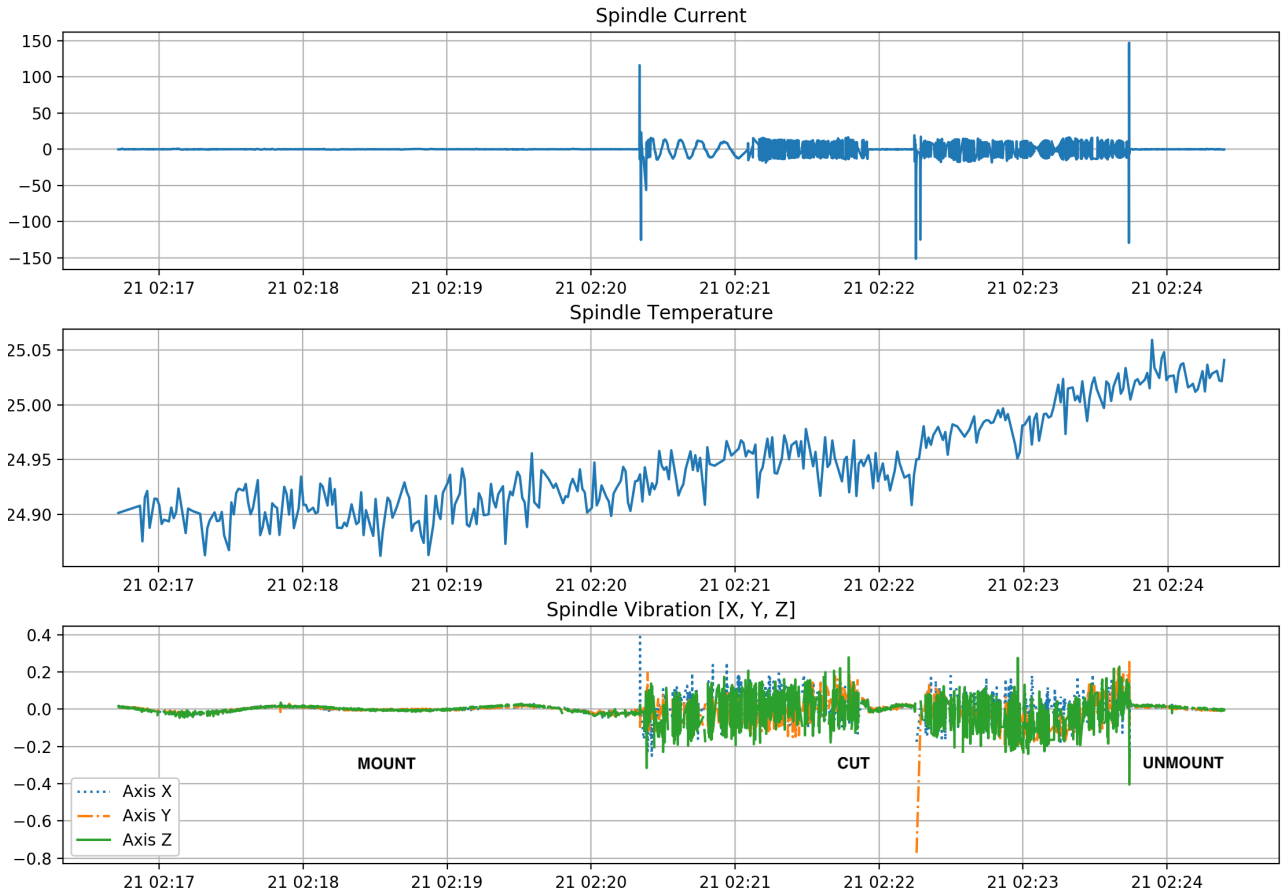

Fig. 5. The sensor readings of the main spindle.

TABLE II
NORMAL BEHAVIOUR (SENSOR PROFILE) OF CURRENT, TEMPERATURE AND VIBRATION SENSOR

Sensor	Stage	Statistical Characteristics		Estimated Reading Range		Drift Trend
		Mean	Std Dev.	Upper	Lower	
Current	MOUNT	0.0781	0.1561	0.2296	-0.1894	0.0000
	CUT	-0.1910	8.3310	15.3485	-15.6400	0.0000
Temperature	MOUNT	24.8901	0.0158	24.9203	24.8618	0.0019
	CUT	24.9440	0.0440	24.9303	24.8694	0.3184
Vibration(x)	MOUNT	0.0033	0.0077	0.0202	0.0103	0.8095
	CUT	0.0064	0.0452	0.0771	-0.0821	0.0000
Vibration(y)	MOUNT	0.0034	0.0071	0.0167	0.0068	0.5780
	CUT	-0.0051	0.0466	0.0645	-0.0715	0.01953
Vibration(z)	MOUNT	0.0064	0.0099	0.0222	0.0090	0.8263
	CUT	-0.0065	0.0568	0.0971	-0.1010	0.0008

B. Dynamic Model Checker

In this section, we demonstrate that a dynamic probabilistic model checker is perceived as a better reflection of the turn-mill machine's behaviour, as compared to a static model. As shown in Fig. 6, a static model was assumed to have a constant confidence score of 0.9 for all sensors. This translated to a seven-day machine failure probability of 1.94E-4, which was constant at all time. With a dynamic model, based on the dataset we collected, there were sensor faults detected in the system which resulted in the degradation of the sensor confidence score. As shown in Fig. 6, this had an effect on the overall system's failure probability, in that the depending on the weight, and the corresponding sensor's confidence score, the failure probability of the machine changed dynamically. We also observed that this has effectively established a strong correlation between the system's failure probability and the sensor's confidence score.

TABLE III
SENSOR CONFIDENCE SCORE OF CUT STAGE

Time	Current	Temperature	Vibration
2020-02-21 10:20:43	0.9543	0.7306	0.7113
2020-02-21 10:21:13	0.9474	0.7424	0.8530
2020-02-21 10:21:43	0.9154	0.6834	0.8540
2020-02-21 10:22:13	0.9547	0.7387	0.8483
2020-02-21 10:22:43	0.9464	0.7180	0.8677
2020-02-21 10:23:13	0.9217	0.6077	0.8747
2020-02-21 10:23:43	0.9381	0.5849	0.7699

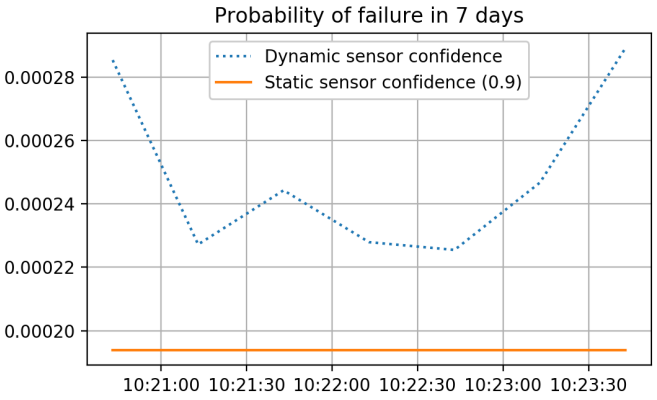

Fig. 6. Comparison of system failure probability.

VI. CONCLUSION

In this paper, we have proposed a new approach integrating a data-driven sensor model and a system-level probabilistic model to form a dynamic probabilistic model. The proposed methodology explicitly models sensor uncertainty and updates expectations on sensor readings with the system under test. The integration in a unified dynamic model provides more accurate prediction results of impending system failures, even while the system is running. However, further research is still needed in the following three areas:

1) The current definition of sensor confidence score is a linear function with each rule having equal weight. This should be generalised so that the proposed approach can be applied to a wider range of systems.
2) The method of consuming sensor confidence scores in the system-level model needs further consideration so that it should represent the real-world systems accurately.
3) A multi-level probabilistic model is needed to represent more sophisticated sensor network-based systems, for example multiple hierarchical modules.

The evaluation results highlight the importance of explicitly modelling sensor trustworthiness, especially because all consequential decisions in the domain of Industry 4.0 will be driven by automatically collected data. Moreover, it helps the system operator to allocate and provision resources timely and efficiently.

REFERENCES

[1] F. Biesinger, D. Meike, B. Kraß, and M. Weyrich, "A digital twin for production planning based on cyber-physical systems: A Case Study for a Cyber-Physical System-Based Creation of a Digital Twin," *Procedia CIRP*, vol. 79, pp. 355–360, 2019. [Online]. Available: https://doi.org/10.1016/j.procir.2019.02.087

[2] E. Negri, L. Fumagalli, and M. Macchi, "A Review of the Roles of Digital Twin in CPS-based Production Systems," *Procedia Manufacturing*, vol. 11, no. June, pp. 939–948, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.promfg.2017.07.198

[3] F. Tao, Q. Qi, L. Wang, and A. Y. 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, 2019. [Online]. Available: https://doi.org/10.1016/j.eng.2019.01.014

[4] A. B. Sharma, L. Golubchik, and R. Govindan, "Sensor faults: Detection methods and prevalence in real-world datasets," *ACM Transactions on Sensor Networks*, vol. 6, no. 3, pp. 1–34, 2010.

[5] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: probabilistic model checking for performance and reliability analysis," *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 4, p. 40, 2009.

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

[7] 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 Transactions on Sensor Networks*, vol. 5, no. 3, pp. 1–29, 2009.

[8] N. Ramanathan, L. K. Balzano, M. Burt, D. Estrin, T. Harmon, C. Harvey, J. Jay, E. Kohler, S. Rothenberg, and M. Srivastava, "Rapid deployment with confidence: Calibration and fault detection in environmental sensor," pp. 1–14, 2006. [Online]. Available: http://escholarship.org/uc/item/8v26b5qh.pdf

[9] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6806 LNCS, pp. 585–591, 2011.

[10] M. Sevegnani, M. Kabáč, M. Calder, and J. McCann, "Modelling and verification of large-scale sensor network infrastructures," *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS*, vol. 2018-Decem, pp. 71–81, 2018.

[11] M. Kwiatkowska, G. Norman, and D. Parker, "Advances and challenges of probabilistic model checking," *2010 48th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2010*, pp. 1691–1698, 2010.

[12] A. Sharma, L. Golubchik, and R. Govindan, "On the prevalence of sensor faults in real-world deployments," *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON*, pp. 213–222, 2007.

[13] D. Alberg and M. Last, "Short-term load forecasting in smart meters with sliding window-based ARIMA algorithms," *Vietnam Journal of Computer Science*, vol. 5, no. 3-4, pp. 241–249, 2018. [Online]. Available: https://doi.org/10.1007/s40595-018-0119-7

[14] I. Turn, "NTX-1000." [Online]. Available: https://en.dmgmori.com/products/machines/turning/turn-mill/ntx/ntx-1000

[15] S.-H. Leitner and W. Mahnke, "OPC UA – Service-oriented Architecture for Industrial Applications," *Softwaretechnik-Trends*, vol. 26, no. 4, pp. 1 – 6, 2006. [Online]. Available: http://www2.cs.uni-paderborn.de/cs/ag-engels/GI/ORA2006-Papers/leitner-final.pdf

[16] R. Richards and R. Richards, "Representational State Transfer (REST)," pp. 633–672, 2006. [Online]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer

[17] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal Aspects of Computing*, vol. 6, no. 5, pp. 512–535, 1994.