# Run-Time Probabilistic Model Checking for Failure Prediction: A Smart Lift Case Study

Xin Xin
*Digital Service*
*TÜV SÜD Asia Pacific*
Singapore
Xin.Xin@tuvsud.com

Sye Loong Keoh
*School of Computing Science*
*University of Glasgow*
Glasgow, United Kingdom
SyeLoong.Keoh@glasgow.ac.uk

Michele Sevegnani
*School of Computing Science*
*University of Glasgow*
Glasgow, United Kingdom
Michele.Sevegnani@glasgow.ac.uk

Martin Saerbeck
*Digital Service*
*TÜV SÜD Asia Pacific*
Singapore
Martin.Saerbeck@tuvsud.com

***Abstract*—Modern smart systems are powered by cyber-physical systems integrating sensor networks with service-oriented architecture to automate their operations. Control algorithms deployed on smart systems are now driven by connected sensors with control decisions being made based on the sensor generated data. As sensors tend to be unreliable and prone to failures, this has resulted in the increase of system errors due to the wrong control decisions derived from the faulty sensor readings, thus affecting the performance, safety and quality of the operational tasks. Existing methodologies to evaluate and test such systems do not take into account the complexity and uncertainty exhibited by the underlying sensor networks, and hence not being able to dynamically verify the behaviour of the smart systems at run-time. This paper proposes a novel run-time verification framework combining sensor-level fault detection and system-level probabilistic model checking. This framework rigorously quantifies the *trustworthiness* of sensor readings, hence enabling formal reasoning for system failure prediction. We evaluated our approach on a passenger lift equipped with sensor networks 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, Sensor Network**

## I. Introduction

There is an increasing number of sensors and actuators being deployed in our environment such as modern smart building systems to provide intelligent controls, predictive analytics and optimisation of energy usage [1]–[4]. With the advent of Artificial Intelligence (AI), machine learning and data analytics, operators are now able to collect large volume of environment and equipment data at run-time, deriving insights on the operations of the building systems. This significantly accelerates the automation of the smart building management system (BMS), allowing for self-control actions to be invoked automatically to ensure energy efficiency and provide fault detection and diagnostics (FDD).

Two important challenges have been identified when deploying a sensor network-based system as the backbone of a smart BMS. Firstly, the system model used to define the operational behaviour of the BMS is typically static and does not accurately reflect the true behaviour of the equipment or sensor over time. It is thus important to identify any deviation to reduce equipment down time and to effectively plan for component replacement and repair. Secondly, there is a strong dependency between the accurate prediction of a 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 lead to the inaccuracy in their predictive maintenance algorithms. We advocate that all maintenance and control decisions are only as good as the sensor data that they are based on.

In real-world deployments, sensor readings typically deviate over time and hence leading to inaccurate readings being acquired. This deviation could be due to wear-and-tear, calibration issues or even intentional data tampering. When these inaccurate readings are used in BMS to derive maintenance schedule, it may lead to errors, degradation of service quality, and more seriously it poses a safety concern to the users. Our observation of the current deployed systems tends to disregard the dynamic nature of the sensor networks and ignoring the inaccurate readings of sensors. Therefore, there is a need to devise a new approach to automate the monitoring of sensor networks that drive the smart BMS. Conventional approaches that are time based, typically involve manual maintenance to test and calibrate sensors using special instruments. Given that there is a large number of sensors driving a wide range of applications, it is no longer feasible to perform periodic manual re-calibration and testing. Hence, there is a strong need for a new sensor network monitoring and testing framework.

We propose a novel run-time verification framework combining data- and model-driven techniques. We introduce the concept of *trustworthiness* to quantify the accuracy and reliability of the sensor readings to allow for the system model to be updated at run-time to reflect the actual behaviour. With the integration of both data- and model-driven approaches, coupled with software testing and cybersecurity principles to ensure data integrity and authenticity, our approach provides a new methodology to model the system behaviour more

X. Xin is partially funded by the Singapore Economic Development Board (EDB) through the Industrial Postgraduate Programme (IPP) Grant.
M. Sevegnani is supported by the EPSRC under PETRAS SRF grants MAGIC and FARM (EP/S035362/1).

accurately over time. We have developed this run-time verification framework to dynamically model a sensor network-based passenger lift and evaluated the performance of this new approach.

The contributions of this paper are as follows:

- 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 status of the smart system over time and predict system failure in a certain period.
- The passenger lifts are safety critical systems in modern smart buildings. This verification framework is validated using a real-life smart passenger lift, involving four sensors and five states.

This paper is organised as follows: Section II provides the background of this research and related work. Section III describes the proposed run-time probabilistic model checking design, with the ability to continuously monitor a sensor network-based system. Section IV presents an implementation to predict the failure probability of a sensor network-based passenger lift. 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

Many technologies and algorithms have been developed to classify sensor faults and verify system formally. *Ramanathan et al.* [5] demonstrated a solution to identify and calibrate a sensor network-based environmental monitoring system. A set of rules were defined to identify sensor faults after they found a significant amount of uninterpretable readings of forty-eight sensors. They showed an efficient method to verify the data integrity of each sensor. However, this method did not reflect the impact of the sensor faults at the system-level.

Three sensor fault categories are systematically classified by *Ni et al.* [6]. This classification is defined according to the nature of sensor attributes, which are *environment features*, *system features or specifications*, and *data features*.

Similarly, *Sharma et al.* summarised four data-driven approaches for detection and identification of sensor faults according to the types of faults, namely *rule-based*, *time series analysis-based*, *learning-based* and *estimation* [7] methods.

Apart from statistical approaches, *Donghyun Park et al.* [8] proposed a data-driven-based light-weight real-time sensor fault detection system. This system employs a Long Short-Term Memory (LSTM) model to detect sensor faults. It overcomes the mathematical approaches limitation which is sensitive to noise and system complexity. However, to train an accurate model is still a challenge in a dynamic environment. Furthermore, the sensor-level fault detection solutions are not capable of quantifying the impact of faulty sensors on the system as a whole.

Probabilistic model checking is a common formal verification technique used to verify a system, analysing the performance and reliability. *Kwiatkowska et al.* [9] demonstrated an application using probabilistic model checking to model a sensor network-based system with probabilistic properties. The resulting model is used to simulate and verify the system's reliability with one or more faulty sensors by injecting failure data. However, this approach lacks adaptation to run-time systems according to sensors' run-time uncertainties. *Calder et al.* [10] introduced another application for failure prediction of a critical communication system. Three status for each component are defined within the system, namely *working*, *reduced-redundancy* and *no-service*. The resulting model is used to evaluate future service availability and predict probability of system failure according to each component's status.

In the interest of verifying a large-scaled sensor network system, *Sevegnani et al.* demonstrated a framework using Bigraphical Reactive Systems (BRS) [11]. This framework is capable of evolving over both the temporal properties and spatial properties, which is the sensor network systems' dynamic behaviour. *Epifani et al.* [12] proposed a novel dynamic probabilistic model checking framework based on Keep Alive Models with Implementations (KAMI). This framework is based on the Bayesian Estimate Theory (BET) to estimate transition matrix according to the run-time system. Subsequently, this estimated transition matrix is applied to a Discrete-Time Markov Chain (DTMC) model to increase the accuracy of failure prediction. All these methods consider the sensor's working condition as a binary result only. However, inaccurate readings of sensors likely affect the system's reliability. And, it is still challenging to involve the quantified trustworthiness of sensor readings in a probabilistic model at run-time.

As the current modelling methods are highly system-specific, manual and domain expertise dependent, significant time and effort are required to build a proper model of the system. *Khoo et al.* [13] proposed a data-driven approach to automatically derive a system's probabilistic model from a historical dataset. This approach provides an alternative method to define the system model without relying much on the domain expert knowledge. It effectively helps to create a programmable system models for complex real-world systems. Even so, this method is not capable to update the probabilistic model dynamically during the run-time.

We are extending existing works by integrating run-time sensor faults detection and quantified sensor trustworthiness so that the probabilistic model can evolve continuously to provide an accurate representation of the physical system's behaviour.

## III. Run-Time Probabilistic Model Checking

We propose an approach to embed quantitative sensor's trustworthiness into the system-level model to predict potential system failure at run-time using the probabilistic model checker PRISM [14] .

A base probabilistic model is first defined according to the system specification to provide a system level abstraction to represent the system behaviour. The initial transition probabilistic matrix of this base-model is manually defined by experienced domain experts. The system then is deployed and starts collecting data to learn about the sensor behaviour and quantify the sensor's *trustworthiness* against expectation. In order to ensure the trustworthiness of the sensor readings, the sensor data is collected and profiled as its *normal behaviour* using time series analysis, estimation and rule-based methods. If any deviation from the norm is detected, the quantified sensor trustworthiness termed as the *confidence* of the sensor will be updated at run-time. For instance, if a sensor fault is detected, the transition probability matrix of the base-model is updated with lower confidence in the trustworthiness of the sensor data. The confidence of sensor trustworthiness leads to the continuous updating of the transition matrix of the base probabilistic model to reflect the system behaviour at run-time. Essentially, the 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. Consequently, this approach takes into account both the run-time sensor readings with quantified sensors' trustworthiness and system-level verification. It allows the verification of the system reliability at run-time and at the same time predicts probability of system failures.

The proposed run-time probabilistic model checking framework [15] employs both the model-driven and data-driven approaches and is implemented by two modules, i.e., model-driven *System Model Verification (SMV)* and data-driven *Sensor Fault Detection (SFD)*. The *System Under Test (SUT)* is the actual sensor network-based system, streaming run-time sensor readings to both SMV and SFD. SFD module calculates the confidence of the trustworthiness of sensor readings and feeds to SMV as an input to update system model. Furthermore, the sensor readings and the system states are saved into a *historical data store* to facilitate the learning of sensor normal behaviour.

### A. Sensor Fault Detection (SFD)

The SFD builds sensor normal behaviour profile from historical data collected over the time. Together with the expert rules and a rule engine, SFD detects and quantifies the confidence of sensor readings at run-time. Two modules are implemented to handle the sensor's data, namely *Sensor Behaviour Analyser (SBA)*, and the *Rules Engine*. SBA establishes the temporal consistency of the sensors. In order to manage the evolution of expectation over time, SBA updates the sensor normal behaviour periodically e.g., on a daily basis. A *reading pipeline* is established and acts a data channel to collect sensor data. In order to derive the *sensor normal behaviour*, SBA obtains the sensor data from pipeline to extract data patterns at run-time.

Each *sensor normal behaviour* comprises three components:

- *Statistical characteristics* — summarises statistic features from a sliding window of sensor readings on an individual sensor basis, e.g. mean, standard deviation, median of each sensor. In the context of sensor fault detection, the *mean* and *standard deviation* are mainly used to measure the reliability of a sensor. For instance, when the standard deviation is higher than its normal behaviour, it adds evidence to be classified as noise [16].
- *Estimation model* — exploits temporal correlation in measurements from historical data-set on individual sensor basis. This model is used to forecast reading range and data patterns.
- *Drift trend* — is the gradient of the trend of normalised sensor readings. This value is computed using an Autoregressive Integrated Moving Average (ARIMA) model to decompose the trend component from sensor readings. Once a consistent deviation is detected, the lower confidence indicates that the reading is less trustworthy. Drift is a typical sensor fault due to wear and tear or calibration errors.

Subsequently, the *Rules Engine* evaluates run-time sensor readings with the *sensor normal behaviour* and domain knowledge to compute the degree of deviation, i.e. sensor's *confidence score*. As the ground truth is not known, this deviation is usually termed as *fault*. In order to quantify the sensor trustworthiness, domain expert knowledge is extracted as a set of pre-configured rules. Afterwards, the run-time sensor readings obtained from the pipeline are assessed according to these rules and the sensor normal behaviour to compute the confidence score. Therefore, the sensor's confidence score is 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.

This *confidence score* is subsequently fed into the SMV (c.f. Section III-B) to update the transition probability matrix of the model at run-time.

### 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 at run-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 defined according to the system specification. They should not change during run-time with only the probability of state transition in the model responding to the working conditions and sensor readings at run-time with changes. The initial probability of state transitions is defined based on the domain expert knowledge with the assumption that all the sensors are fully trusted. Subsequently, the transition probability matrix $P$

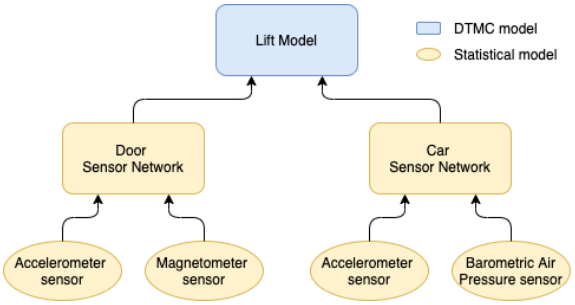


Fig. 1. The sensor networks of a passenger lift.

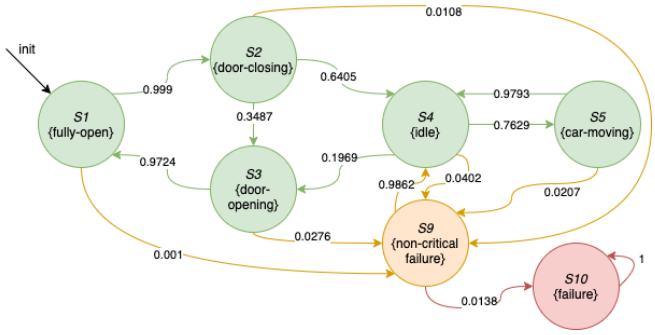


Fig. 2. The state transition model of a passenger lift.

evolves over time in keeping with the sensor's trustworthiness, which is the confidence score obtained from the SFD. With this approach, the resulting model reflects the real behaviour of the system according to the confidence of sensors, thus leading to more accurate prediction results and better verification performance.

## IV. Experimental Design

We demonstrate through an experiment illustrating how the sensor's confidence score is computed and influences the probability model of a passenger lift system in this section.

Modern passenger lifts are equipped with sensors to capture and process real-time data to monitor load conditions, detect abnormal behaviour and to estimate when maintenance should be performed. The efficiency and accuracy of these maintenance processes are based on the quality of sensor readings.

We modelled a passenger lift using the proposed run-time probabilistic model checking framework to evaluate this approach's performance. Internal institutional lift experts provided a list of lift parameters to be monitored, with the lift motion status and door states having the highest priorities. Hence, we set up two sensor networks for a passenger lift, one monitors the door, while the other is responsible for the car movement. The door module comprises two sensors, namely *accelerometer* sensor and *magnetometer* sensor. The car module includes *accelerometer* sensor and *barometric air pressure* sensor. Fig. 1 shows the hierarchical structure of the lift model. These sensors are connected to a gateway located in the lift cabin, the extracted sensor readings are streamed to the back-end data processing pipeline via *Advanced Message Queuing Protocol (AMQP)* [17].

### A. Sensor Fault Detection (SFD) Module

SFD was implemented using Python and the interfaces were developed following RESTful [18] architecture patterns. Historical data and configurations are provided through a web-based interface. SFD inspects historical readings to extract sensor normal behaviour on a monthly basis. During run-time, the sensor readings are generated by the lift and published to a cloud-based AMQP service. With this, all processing modules can subscribe to the service to obtain the data to be processed in a five minutes interval. SFD receives the run-time readings to compute a *confidence score* for each sensor against its normal behaviour. Afterwards, this confidence score is fed to SMV to update the transition probability 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 derived according to [13]. Thereafter, each state is labelled by domain experts based on the domain knowledge and operating experiences. We describe an illustrative case to evaluate the proposed approach. Fig. 2 shows the lift model, which includes five working states and two error states:

1) *fully-open* — The doors of the passenger lift are fully opened. As advised by the operators, this is the initial state of the passenger lift. The sensors' readings are most stable at this stage.
2) *door-closing* — The doors start to close until they are fully closed. At this stage, any interruption to the door movement will result in the doors turning to open again. The lift's state is then moved to *door-opening*. During this stage, the door's sensor network readings reflect the movement behaviour. However, the car sensor network's readings remain stable.
3) *door-opening* — The doors start to open until they are fully opened. During this stage, the door's sensor network readings reflect the movement behaviour, but the car sensor network's readings remain stable.
4) *idle* — The doors are fully closed and the lift cabin car is stationary. During this stage, both door and car sensor networks' readings are stable.
5) *car-moving* — The doors are fully closed and the lift cabin car is moving. During this stage, the door sensor network's readings are stable, while the car sensor networks' readings demonstrate the moving behaviour.
6) *non-critical failure* — The lift is working in an unexpected condition. However, it can be recovered automatically.
7) *failure* — This is a state that indicates the lift is working with unexpected behaviour that may cause subsequent hazard or injury.

and the initial transition probability matrix is defined based on Fig. 2. The initial transition probability matrix, assumes that the sensors are new and behave correctly. The probability of system failure is expressed by the following Probabilistic Computation Tree Logic (PCTL) [19] formula:

$$P_{failure} =? \left[ F^{\leq 24\times 30} \left( S_{10} \right) \right] \tag{3}$$

where $P_{failure}$ is the probability of lift failure in thirty days. The failure state $S_{10}$ is defined in the lift model in Fig. 2.

Ideally, the model should accurately reflect the system state. In practise, this is not always the case. As mentioned, existing methods tend to assure the sensors are working correctly at all time. In reality, the deviation is bound to occur, among others, by wear and tear of the sensors or sensor readings drift. This deviation causes inaccurate decisions to be made by the control system. In order to make the system model to reflect the real behaviour, the probability of each transition can be updated according to the run-time confidence score of each sensor from the SFD.

As each lift state is monitored by two sets of sensors as presented in Fig. 1 and each set is independent from the other, we aggregated one set of sensors as a sensor network. With this, we calculated a sensor network confidence score as a whole to represent the set of the sensors. In this lift context, we calculated the confidence scores of two sensor networks, i.e., the *door network* and *car network* as following:

$$C_{door} = \{C_m \times w_m\} + \{C_a \times w_a\} \tag{4}$$

$$C_{car} = \{C_b \times w_b\} + \{C_a \times w_a\} \tag{5}$$

where the $C_m$, $C_a$ and $C_b$ are the confidence scores of the magnetometer, accelerometer and barometric sensor respectively. The $w_m$, $w_a$ and $w_b$ are the weight assigned to the sensors in the current network. Subsequently, the probability of each transition can be updated according to the run-time sensor network confidence score based on the following:

$$P_{rt} = \begin{cases} P_{init} \times C_{door} & state \in \{S1, S2, S3\} \\ P_{init} \times \{C_{door}, C_{car}\} & state \in \{S4\} \\ P_{init} \times C_{car} & state \in \{S5\} \end{cases} \tag{6}$$

where the run-time probabilistic of transition between states depends on the confidence scores of two sensor networks, $C_{door}$ and $C_{car}$ are sensor networks' confidence scores obtained from Eq. 4 and Eq. 5, $P_{init}$ is the initial probability matrix.

Consequently, by knowing $P_{rt}$, the system's initial lift probabilistic model, $P_{init}$ can be dynamically updated whenever sensor faults are detected. The confidence coefficients are defined based on a domain expert's experience for this specific lift model. Hence, domain expertise is not only used for the initial model, but also guides the update of expectations.

### C. Assumptions

The proposed run-time probabilistic model for the passenger lift was implemented based on the assumptions defined below:

1) The initial transition probability matrix, $P_{init}$ was first derived using Evidence-Driven State-Merging (EDSM) algorithm [20]. Subsequently, the experienced operators fine-tuned this matrix accordingly to actual running status.
2) An experienced operator sets the weights of all sensors, i.e., the weights of sensors in the door sensor network:
   a) Magnetometer sensor: 0.6
   b) Accelerometer sensor: 0.4

   The weights of sensors for the car sensor network:
   a) Barometric air pressure sensor: 0.5
   b) Accelerometer sensor: 0.5
3) The initial *sensor confidence score* is set as 0.99 for all individual sensors.

## V. Experimental Results

We observed the five lift states, *fully-open, door-closing, door-opening, idle* and *car-moving* to evaluate our implementation.

### A. Sensor Fault Detection (SFD)

In this implementation, we grouped the sensors into two sensor networks: door network and car network. Each sensor network's normal behaviour was calculated based on the individual sensors according to equation 1, 4 and 5. For the confidence score of sensors and sensor networks in each lift state, the same algorithm (c.f. Section III-A) was used to analyse the readings.

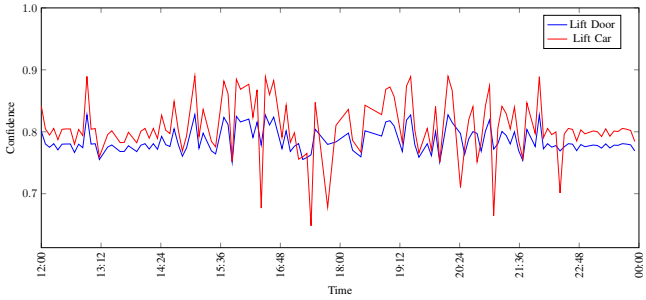


Fig. 3. Sensor Network confidence score of Lift-Door and Lift-Car.

We configured the rules in consonance with the domain of lift management to validate the concept. The actual parameter values are related to the individual lift's characteristics and hence were chosen conservatively. They are generally fine-tuned during the deployment stage to improve the performance. The following example rules are sufficient to evaluate the methodology:

1) If the lift status is not *idle*, but there is no variation (standard deviation) in the sensor readings for more than thirty seconds, the sensor is considered as having *Stuck At* fault.
2) According to the experiences, in case that more than 50% of the readings are missing in the window of five minutes, the sensor is deemed as having an *Intermittent*

fault. Otherwise, the normalised ratio of the received readings and the total number of expected sensor readings are returned.

3) Compute the distance between the actual sensor reading pattern and the sensor's normal behaviour model. This distance is used as the factor to detect *out of range* faults.
4) If the drift trend value is greater than 0.5 which should be close to 0, the sensor is deemed as having a *Drift* fault.

Fig. 3 shows a snapshot of two sensor-networks' confidence score of a working day operation. Two series shows a similar pattern that they are more stable in the morning than the afternoon. This means that there were more rides in the afternoon in this commercial building. However, the car sensor-network presents higher fluctuation. In this case, the operator need to pay more attention to inspect either the lift car sensor-network or the related components, e.g., electric motor, pulleys and metal cables, etc. It is relatively straightforward to identify the working condition of each sensor according to the confidence score. However, it is quite challenging to determine the impact of the lift's overall operating situation in line with the sensor's working condition. With the proposed run-time probabilistic model checking approach, the resulting confidence scores are employed to update the transition matrix of the model at run-time.

### B. Run-time Model Checking

This section demonstrates how run-time probabilistic model checking presents a better reflection of the passenger lift's behaviour compared to a static model. Fig. 4 shows a static model that assumed all sensor networks retain a steady confidence score of 0.9. The resulting model provided a thirty-day's failure probability of 0.0432, which was constant at all time. In the actual situation, based on the dataset we collected, sensor faults were detected. It resulted in the degradation of the sensor confidence score, as shown in Fig. 3. This affected the overall system's failure probability. Fig. 4 shows the failure probability of the passenger lift changed with a model updated at run-time. A strong correlation was established between the system's failure probability and the sensor-networks' confidence score effectively.

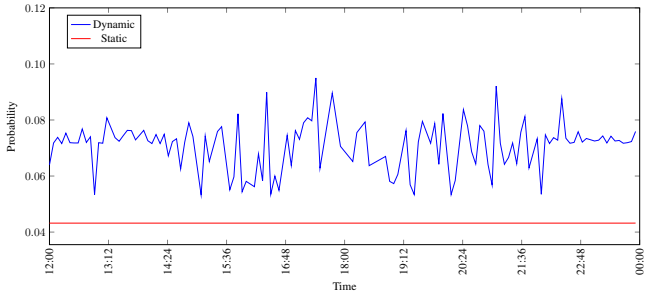


Fig. 4. Comparison of system failure probability.

## VI. Discussion

We have proposed a novel sensor-network verification framework for sensor network-based smart systems using passenger lift as an evaluation use case. This framework can be used to predict potential system failure probability over time, model the sensor network's behaviour and quantify the trustworthiness of a sensor network at run-time.

With the passenger lift use case, we managed to derive an abstract lift model comprising two sensor networks which are responsible for monitoring lift doors and lift cabin car respectively. In order to reflect the run-time sensor network's behaviour, we set up a data processing pipeline to collect sensor readings and calculate statistical characteristics, including total readings received, the mean value of the readings of each lift state and standard deviations. Furthermore, we developed a batch job to compute the sensor-level-drift trend periodically, e.g. monthly basis. By combining the statistical methods and drift-trend, our verification framework is able to quantify the trustworthiness of a sensor network, namely the confidence score. Subsequently, this confidence score is fed to a probabilistic lift model to compute the probability of a system failure over thirty days through model checking.

Verifying the passenger lift using the proposed approach has provided insightful findings. We provide further analysis and discussion of the experiment results and findings below:

1) The system failure probability is in inversely proportion to the sensor's confidence score. When the sensor's confidence score is low, this results in the high probability of a failure. In the case of the passenger lift, a linear function is applied to compute the sensor network confidence score to the transition probability matrix. The choice of function very much depends on the logical relationship between the sensors and the system, this can be configured and customised accordingly. Extensive discussions were held with the passenger lift domain experts and it is appropriate to define a linear function for this use case. Nonetheless, machine learning might be another option if the necessary dataset is available.
2) As the sensor is dynamic in nature, a transient fault in the sensor leading to the system failure will recover automatically. Thus, we observed that the overall system failure probability fluctuated over time, and did not show a trend towards high failure probability. For instance, when unexpected interference happened, this caused the abnormal fluctuation of the sensor readings. Whenever this interference has stopped, the environment is back to normal, which means that the sensor recovered and it is back to normal behaviour as well. Naturally, the overall system's failure probability will be reduced accordingly. In fact, some recoverable failures were captured, e.g, an obstacle blocked the cabin doors which had led to door closing failure. Once the obstacle had been removed, the lift recovered automatically. We used a general state named *non-critical failure* as in Fig. 2 to capture these

transient faults, with the assumption that there is a high probability of recovery.

3) Throughout the 2-month monitoring of the passenger lift, we did not observe any persistent sensor failure and that no actual lift failure had occurred. Hence, the resulting system failure probability was rather low. This could be attributed to the frequent maintenance and the replacement of sensors to ensure the reliability of the lift. With the proposed verification framework, we can optimise the maintenance schedule by examining the system failure probability such that when a threshold is reached, it triggers the maintenance process to take place.

The experiment result shows a strong correlation between the sensor network's trustworthiness and the probability of system failure. Considering the dynamic nature of the sensor networks, this framework provides a viable approach for monitoring and maintaining sensor network-based systems, especially for safety-critical systems.

## VII. Conclusions

We have demonstrated a new approach that combines data-driven sensor model for quantifying the sensor trustworthiness and model-driven probabilistic system abstraction to form a rum-time probabilistic model. We described how to update a probabilistic model of sensor network 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 three directions:

1) The rules for deriving sensor network confidence score are manually defined according to the expert on a specific system basis. This should be generalised so that the proposed approach can be applied to a wider range of systems.
2) A context-awareness adaptive algorithm is needed to reflect more general scenarios, e.g., the lift moves with passengers and without passengers.
3) A hybrid probabilistic model is required to model more sophisticated sensor network-based systems. For instance, to model control system as discrete model and physical modules as continuous model.

The evaluation results have highlighted the efficiency of explicitly modelling sensor trustworthiness, especially because all consequential decisions of sensor network-based systems will be driven by automatically collected sensor data. Moreover, it helps the system operators allocate and provision resources timely and efficiently.

## References


[1] A. Shah, H. Nasir, M. Fayaz, A. Lajis, and A. Shah, "A Review on Energy Consumption Optimization Techniques in IoT Based Smart Building Environments," *Information*, vol. 10, no. 3, p. 108, Mar. 2019.

[2] H. Zou, Y. Zhou, H. Jiang, S.-C. Chien, L. Xie, and C. J. Spanos, "WinLight: A WiFi-based occupancy-driven lighting control system for smart building," *Energy and Buildings*, vol. 158, pp. 924–938, Jan. 2018.

[3] Y. Agarwal, B. Balaji, R. Gupta, J. Lyles, M. Wei, and T. Weng, "Occupancy-driven energy management for smart building automation," in *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building - BuildSys '10*. Zurich, Switzerland: ACM Press, 2010, p. 1.

[4] H. Park and S.-B. Rhee, "IoT-Based Smart Building Environment Service for Occupants' Thermal Comfort," *Journal of Sensors*, vol. 2018, pp. 1–10, 2018.

[5] N. Ramanathan, L. 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 Networks," 2006.

[6] K. Ni, N. Ramanathan, M. N. H. Chehade, S. Nair, S. Zahedi, G. Pottie, M. Hansen, M. Srivastava, and E. Kohler, "Sensor Network Data Fault Types," vol. 5, no. 3, p. 29.

[7] 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. 1–39, Jun. 2010.

[8] 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.

[9] M. Kwiatkowska, G. Norman, and D. Parker, "Quantitative Analysis With the Probabilistic Model Checker PRISM," *Electronic Notes in Theoretical Computer Science*, vol. 153, no. 2, pp. 5–31, May 2006.

[10] 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, Jan. 2019.

[11] M. Sevegnani, M. Kabac, M. Calder, and J. McCann, "Modelling and Verification of Large-Scale Sensor Network Infrastructures," in *2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS)*. Melbourne, VIC: IEEE, Dec. 2018, pp. 71–81.

[12] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli, "Model evolution by run-time parameter adaptation," in *2009 IEEE 31st International Conference on Software Engineering*. Vancouver, BC, Canada: IEEE, 2009, pp. 111–121.

[13] T. P. Khoo and J. Sun, "The Miles Before Formal Methods - A Case Study on Modeling and Analyzing a Passenger Lift System," in *Formal Methods and Software Engineering*, J. Sun and M. Sun, Eds. Cham: Springer International Publishing, 2018, vol. 11232, pp. 54–69, series Title: Lecture Notes in Computer Science.

[14] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, ser. LNCS, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 585–591.

[15] X. Xin, S. L. Keoh, M. Sevegnani, and M. Saerbeck, "Dynamic Probabilistic Model Checking for Sensor Validation in Industry 4.0 Applications," in *2020 IEEE International Conference on Smart Internet of Things (SmartIoT)*. Beijing, China: IEEE, Aug. 2020, pp. 43–50.

[16] A. Sharma, L. Golubchik, and R. Govindan, "On the Prevalence of Sensor Faults in Real-World Deployments," in *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, Jun. 2007, pp. 213–222, iSSN: 2155-5494.

[17] S. Vinoski, "Advanced Message Queuing Protocol," *IEEE Internet Computing*, vol. 10, no. 6, pp. 87–89, Nov. 2006.

[18] R. Richards and R. Richards, "Representational State Transfer (REST)," pp. 633–672, 2006.

[19] 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.

[20] S. Verwer, M. de Weerdt, and C. Witteveen, "Efficiently identifying deterministic real-time automata from labeled data," *Machine Learning*, vol. 86, no. 3, pp. 295–333, Mar. 2012.