Software Quality Journal (2022) 30:367–388
https://doi.org/10.1007/s11219-021-09559-w

Learning and analysis of sensors behavior in IoT systems using statistical model checking

Salim Chehida[1] · Abdelhakim Baouya[1] · Saddek Bensalem[1] · Marius Bozga[1]

Accepted: 28 April 2021 / Published online: 18 June 2021

Abstract
Analyzing the behavior of sensors is becoming one of the key challenges due to their increasing use for decision making in IoT systems. The paper proposes an approach for a formal specification and analysis of such behavior starting from existing sensor traces. A model that embodies the sensor measurements over time in the form of stochastic automata is built, then temporal properties are fed to Statistical Model Checker to simulate the learned model and to perform analysis. LTL properties are employed to predict sensors' readings in time and to check the conformity of sensed data with the sensor traces in order to detect any abnormal behavior. We also use LTL properties to analyze the collective behavior of a set of sensors and build a formal model that checks the conformity of a combination of sensors' readings in time.

Keywords IoT · Sensor Behavior · Stochastic Automata · Statistical Model Checking · LTL · BIP

1 Introduction

Internet of Things (IoT) has become one of recent technology mostly used in various domains such as health and environmental monitoring (Tao, 2020), construction and energy management (Park et al., 2018), smart vehicles (Al-Turjman & Malekloo, 2019) and buildings (Daissaoui et al., 2020). It consists of a collection of entities that interacts with users to fulfill a common goal. The sensor is a critical device in the IoT ecosystem that allows to measure the state information over time and monitor physical components. Data gathered from different sensors are used to make a decision and promote automation in IoT systems by providing efficient and intelligent services, whereas corrupted data during transmission or malfunction of sensors, due to natural events or other causes can influence the correct operation of the entire system. Indeed, the massive increase of these issues with the growing number of deployed sensors pushes toward the sensors' behavior analysis by checking their sensed data.

✉ Salim Chehida
Salim.Chehida@univ-grenoble-alpes.fr

[1] University of Grenoble Alpes, CNRS, VERIMAG, Grenoble F-38000, France

The analysis of sensors' behavior and detecting the erroneous readings have attracted great attention. Many approaches have been proposed based on several methods such as *statistical methods* (Yu et al., 2014), *probabilistic methods* (Hill et al., 2009; Xie & Shun-Zheng, 2009), *clustering-based methods* (He et al., 2003) and *prediction-based methods* (Shahid et al., 2015). Governed by the standard learning requirements, the approaches rely on the metadata and structure of the sensed data. In this paper, we propose a model-based approach involving formal specification for sensors' behavior analysis. Our approach aims to make the analysis process rigorous, automatic, scalable and meaningful. In our approach, we start by collecting sensors traces and data preprocessed required to build an approximate model of the sensors' behavior, then we apply formal verification techniques to analyze the learned models and check if new measurements are compliant with these models. Although our approach cannot be used to detect the causes of abnormal behavior of sensors, it can however help to predict sensors' readings and identify possible abnormal readings based on past observations.

Model checkers allow checking the conformity of a system model expressed in formal notation to a set of properties expressed in a logical language. In this study, we apply a type of model checkers called *Statistical Model Checkers* (SMC) to verify whether a sensor model expressed in Stochastic Automata (SA) satisfies a given logical property up to some probability, based on model simulations. We use *quantitative* properties expressed by Linear-time Temporal Logic (LTL) to predict the sensor readings in time and *qualitative* LTL properties to check the confidence of sensed data and their compliance with the provided traces. Several SMC tools have been proposed such as PRISM-SMC (Kwiatkowska et al., 2011) and UPPAAL-SMC (David et al., 2015a). The BIP language (Basu et al., 2011) and *S*BIP (Mediouni et al., 2018) are used in this paper for behavior modeling and SMC analysis. BIP allows the rigorous design of component-based systems. The choice of BIP is also motivated by its capability for specifying stochastic behaviors and their analysis based on SMC using the *S*BIP tool.

We apply our approach to the industrial case study of the Cecebre dam in Spain, which is equipped with wireless sensors that measure the water contributions to the dam. There are three types of sensors that are used to measure the *Water Height* (*WH*), the *Rain Precipitation* (*RP*) and the *Water output Flow* (*WF*). As shown in Fig. 1, the data collected from sensors are used to control the opening of the spillgate in order to ensure that the water does not reach a maximum level in the dam. The anomalous behavior of these sensors can influence the correct operation of the dam system. Our objective is to build formal models that specify and analyze the behavior of the sensors by checking the flow of data produced by these sensors. A trace of time series data recorded by each sensor per day from 1989 to 2016 has been collected. We reorganized the original trace by creating a separate CSV file per sensor. Each file contains one sensor readings per day for 28 years. The collected data will be used to build the sensor behavior model.

This paper enhances and extends the approach presented in (Chehida et al., 2020) by assembling the behavior models of a set of sensors and expressing LTL properties for the analysis of their collective behavior. Analyzing the collective behavior of sensors can give a more complete image of the phenomenon under observation and reveals interesting consistency between sensors' behaviors.

The paper is organized as follows: we give an overview of our approach with the tools that support it in Section 2. We build the sensors' behavior models in Section 3. The analysis results per sensor and of the collective behavior will be presented in Sections 4 and 5. Finally, we present related works in Section 6 and draw our conclusions in Section 7.

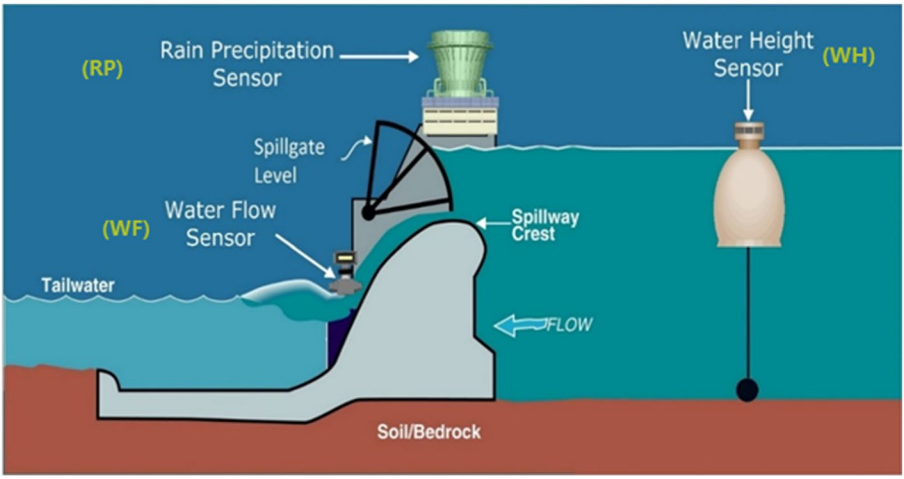

Fig. 1 Dam Infrastructure

2 Overall Approach

Figure 2 shows the main steps of our approach that combines data-driven and component-based paradigms. Starting from the trace of sensed data, we build component models that learn the behavior of sensors, then we analyze these models using statistical model checking.

Collection and preprocessing of sensor data. In the first step, we start by collecting and preparing the sensors data described as time series that contain successive measurements of sensors in time. Data preprocessing is a fundamental activity in numerous computer science fields, such as machine learning, data mining and pattern recognition. It consists of removing errors in original data and preparing the data in a suitable and useable format, which leads to quality analysis and learning results. As shown in Fig. 3, the main tasks of sensor data preprocessing in our approach are *data cleaning* and *data discretization*.

- The data cleaning searches for and then removes or repairs errors and inconsistencies in sensors data.
- The data discretization converts continuous (or quantitative) data into discrete (or qualitative) ones. It aims to reduce the number of values for continuous time series by dividing sensors' readings into intervals. Discrete values are easy to use and understand, which facilitates learning and analysis of sensors' behavior. (Yang et al., 2010) presents the several methods proposed for time series data discretization. In this study, we use the EWD (Equal Width Discretization) method (Dougherty et al., 1995) because of its simplicity.

Fig. 2 Generic Approach for Sensor Behavior Analysis

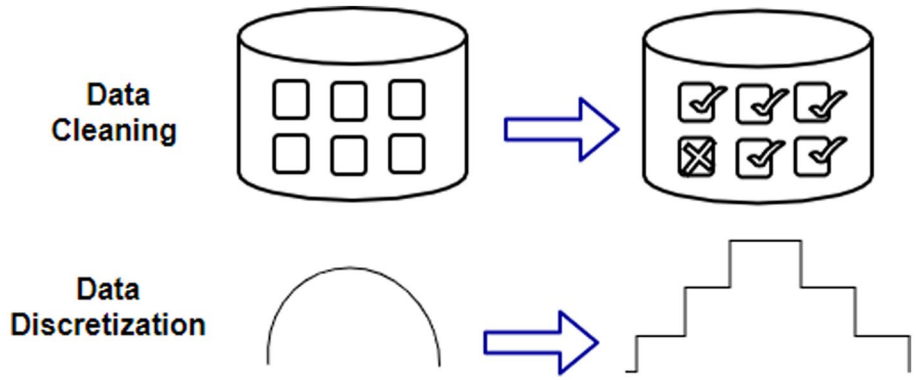

Fig. 3 Data Preprocessing Steps

Specification of sensor behavior model. In the second step of our approach, we use the BIP language[1] for modeling the sensors' behavior as automata starting from preprocessed sensors data. BIP (Behavior, Interaction, Priority) is a highly expressive component-based language for the rigorous design of complex systems. It allows representing the behavior of systems using a set of components, a set of interactions that defines the possible communications between the components and a set of priorities for defining interaction schedule policies. Figure 4 presents the main concepts of BIP and their relationships. Atomic components, called *Atoms*, are the simplest component type (i.e., non-hierarchical) whose behavior is expressed by finite-state automata (see example of Fig. 7). BIP supports the formal modeling of stochastic systems based on Discrete and Continuous Time Markov Chains (DTMC and CTMC) and Generalized Semi-Markov Process (GSMP). Automata have transitions labelled with ports and states that denote control locations where component waits for interactions. Ports are actions that can be associated with data stored in local variables and used for interactions with other components. Types of variables are either

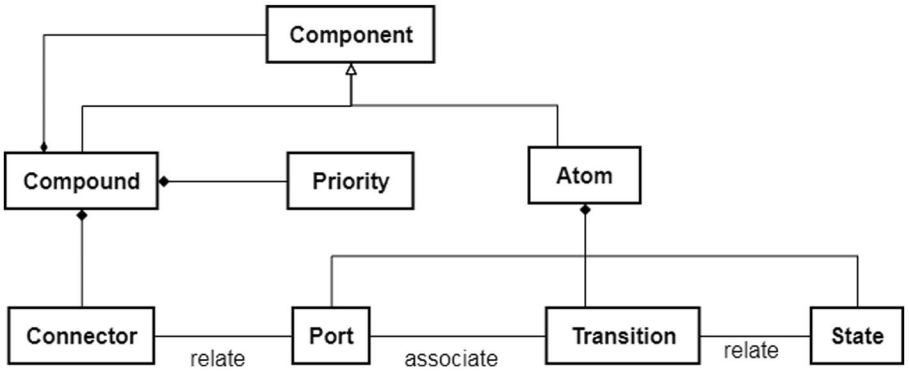

Fig. 4 BIP Concepts

[1] https://www-verimag.imag.fr/TOOLS/DCS/bip/doc/latest/html/index.html

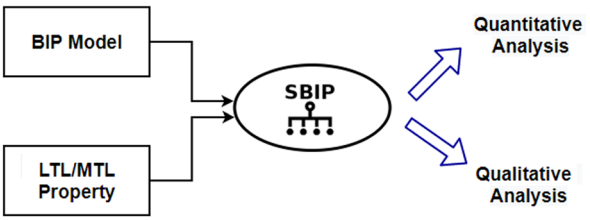

Fig. 5 Analysis of BIP Models with *S*BIP Statistical Model Checker

native (Boolean, integer, float or string) or external (to be externally defined such as lists or files). As for external types, the BIP language allows the declaration of external function prototypes that are assumed to be externally defined using programming language like *C++*. BIP supports the specification of composite, hierarchically structured components called *Compounds* starting from the atomic ones. A compound is composed of a set of components, connectors and priorities (see example of Fig. 16). Connectors relate ports from components by assigning to them a synchronization attribute, which may be either trigger or synchronous. Priorities are used to favor the execution of a subset of enabled interactions. They can be used to resolve the conflict between interactions or to express particular scheduling policies.

Analysis of sensor behavior. In the final step of our approach, we use *S*BIP framework[2] for the analysis of sensors' behavior expressed by BIP. *S*BIP has a graphical user-interface permitting to edit, compile and simulate models, and automates the different statistical analysis. As shown in Fig. 5, the input of the tool is a system model S expressed in BIP language like that of Fig. 7 and a property ϕ expressed in Linear-time Temporal Logic (LTL) (Pnueli, 1977) and/or Metric Temporal Logic (MTL) (Alur & Henzinger, 1993). Using *S*BIP, we can perform two types of analysis:

1. *Quantitative*: we estimate the probability that the system S satisfies a given property ϕ.
2. *Qualitative*: we test whether the probability of a given property ϕ being satisfied by the system S is greater or equal to a certain threshold θ.

To decide whether S satisfies ϕ (written $S \models \phi$), *S*BIP refers to simulation-based techniques: *Probability Estimation* (PE) (Hérault et al., 2004) for quantitative properties and *Hypothesis Testing* (HT) (Younes & Simmons, 2002) for qualitative properties. PE computes the probable values of the parameters based on a given distribution and HT determines the extent to which the observations meet a given property. In both techniques, the answer given will be correct up to a certain level of confidence.

3 Modeling sensor behavior

We first start by data preprocessing and extraction of some statistical information needed to build the behavior models of sensors.

[2] http://www-verimag.imag.fr/BIP-SMC-A-Statistical-Model-Checking.html?lang=en

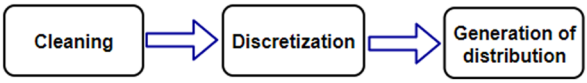

Fig. 6 Preprocessing of Sensor Data

3.1 Data preprocessing

As shown in Fig. 6, we define the following steps for data preprocessing of the raw data collected by the sensors and generating the distribution file (*sensorDistribution*) for each sensor :

- *Cleaning*. The dataset provided by our industrial partner contains the raw data reported by sensors. In the first step of cleaning, we use a filter to search and remove the erroneous sensors data. The filter removes the values outside of the expected range of data such as NaN, negative values and the values that are outside the boundary of the normal interval. For example, the normal interval of *WH* sensor is between 28 m and 36 m. In the second step, we use a filter to check and eliminate inconsistencies in consecutive sensors data. For instance, the difference between the two values recorded by *WH* sensor in two consecutive days could not exceed 1 m. The unique case of violation of this rule is identified in data collected in 1994: the value reported in January 7 (31, 54) exceeds the value reported in January 6 (30, 50) by more than one meter and has been removed. We note that few errors and inconsistencies are found in the initial dataset.
- *Discretization.* In this step, we use the EWD (Equal Width Discretization) method (Dougherty et al., 1995) for mapping numerical values into predefined fixed intervals that have an equal-width. Each bin or level is associated with a distinct discrete value. In this work, we relied on data visualization using histograms to determine the number of levels. The discretization of *WF*, *RP* and *WH* sensors data is stated into five levels. Our approach can be used to build models with arbitrary number of levels (not necessary five). However, for sensors data considered in our case study, this number was sufficient. There also other methods that can help for determining a suitable number of bins (Alvarez et al., 2013).
- *Generation of distribution.* In this step, we extract some statistical information once data was discretized. We use the classical statistical function called PMF (Probability Mass Function) (Stewart, 2009) that assigns a probability for specific discrete values. PMF is often the main way to define a discrete probability distribution for scalar or multivariate random variables whose domain is discrete. Using PMF, we generate a sensor distribution file (*sensor Distribution*) that defines the probabilities of sensor readings levels by counting the occurrence of each level of sensor readings each day.

The sensor distribution defined using classical statistical methods (EWD and PMF) is the primary means for specifying the sensor behavior model.

3.2 BIP behavior model

In our method, we build a BIP component for every type of sensor. Figure 7 presents an example of a behavior model for the water height sensor expressed as *Stochastic Automata* (SA) using the BIP language. The stochastic semantics is defined by variables based on the probability distributions. BIP supports discrete distributions such as *sensorDistribution* and also standard distributions, such as *Uniform*, *Normal* and *Exponential*.

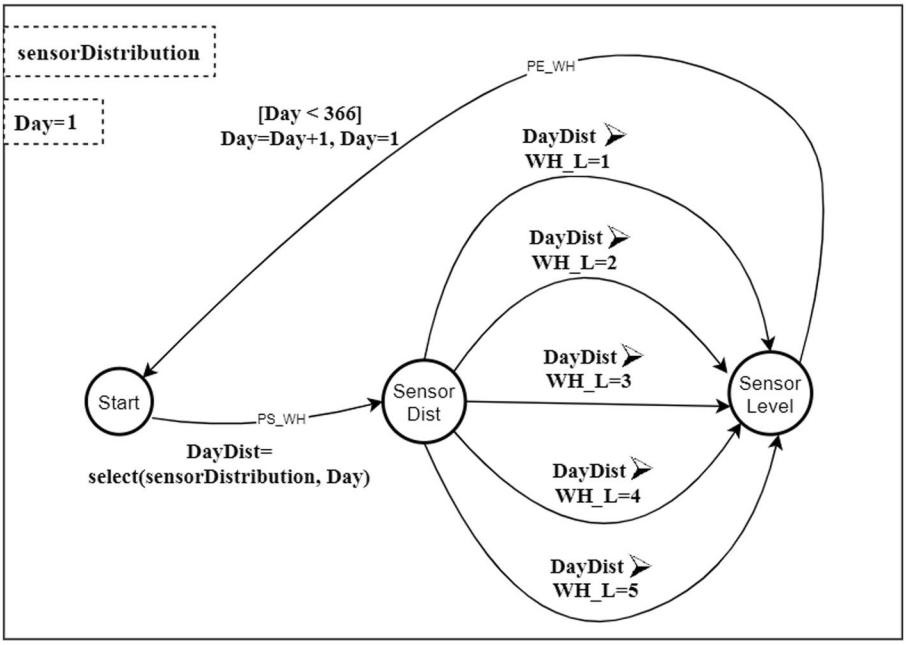

Fig. 7 BIP Behavior Model of Water Height Sensor

In the model of Fig. 7, the port *PS_WH* selects the day distribution based on *sensor distribution* file generated in the previous section. According to this distribution, the water high level (1, 2, 3, 4, or 5) is defined based on the variable *DayDist*. The port *PE_WH* increments the variable *Day* and starts a new iteration.

The models that specify the behaviors of the other sensors (*RP* and *WF*) are defined using the same pattern as *WH* sensor model. Only the sensors' distributions can change depending on the trace of sensors data. These distributions will be updated with new sensors' observations. The modeling approach can be used to represent the flow of data produced for other time scales. For example, the values generated every hour in the day.

Our component-based approach facilitates the reusability and maintainability of the system components. A new BIP components can be defined if the system incorporates new sensors or replaces existing ones. Also, we can assemble certain sensors components to analyze their collective behavior (see Fig. 16).

Using the built models, we can simulate and analyze the behavior of each sensor for any period of the year (Section 4). We can also analyze the collective behavior of sensors (Section 5).

4 Analysis of sensor behavior

In this work, we use a stochastic bounded variant of LTL to express properties. LTL is the natural choice in the context of runtime monitoring and runtime verification (i.e., using statistical model checking), where properties are expressed and evaluated on traces. In LTL,

path formulas are defined using four bounded temporal operators namely, `Next` ($N\psi_1$), `Until` ($\psi_1 \cup^k \psi_2$), `Eventually` ($F^k\psi_1$), and `Always` ($G^k\psi_1$), where k is an integer value that specifies the length of the considered system execution trace and ψ_1,ψ_2 are called state formulas, which is a Boolean predicate evaluated on the system states.

4.1 Quantitative analysis

We present four examples of quantitative properties expressed on the model of Fig. 7. The properties can be parametrized by the day of the year. Any day or interval of days can be chosen for evaluation. *S*BIP allows to check parametric property ϕ(x), where x is a parameter ranging over a finite instantiation domain. It also provides a summary of the performed analysis and generates specific curves and/or plots of results.

Property 1: The probabilities of obtaining each level (L=1..5) from the water height sensor on April 27.

In LTL:

$P_{=?}[F^{3000}\ (WH\_L = L\ \&\&\ Day = 118)];\ \ L = 1 : 5 : 1;$

The results are given in Fig. 8. We find that level 5 is the most likely and levels 4 and 3 are less likely. However, levels 1 and 2 are never observed on this day. These predictions concerning water height sensor and estimations from other sensors can help the managers of dam infrastructure to adjust the spillgate level.

Property 2: The probabilities of obtaining each level (L=1..5) from the water height sensor at the first weeks of January and May.

In LTL:

$$\begin{cases} P_{=?}[F^{3000}\ (WH\_L = L\ \&\&\ Day = T)];\ \ T = 1 : 7 : 1; T = 122 : 128 : 1; \\ \quad L = 1 : 5 : 1; \end{cases}$$

Figure 9 shows the SMC verdict of property 2. We see that level 5 is rarely observed in the first week of January; however, this level is most likely in the first week of May. The opposite for levels 1 and 2, which are more possible in the first week of January and rare in the first week of May. With LTL properties, we can predict the evolution of water height level at any period of the year.

Property 3: The probabilities that each level (L=1..5) obtained from the water height sensor remains the same at the last week of May.

In LTL:

$$\begin{cases} P_{=?}[G^{3000}\ (WH\_L = L\ \&\&\ Day = 146) \cup^{3000}\ (WH\_L = L\ \&\&\ Day = T)]; \\ T = 147 : 152 : 1;\ L = 1 : 5 : 1; \end{cases}$$

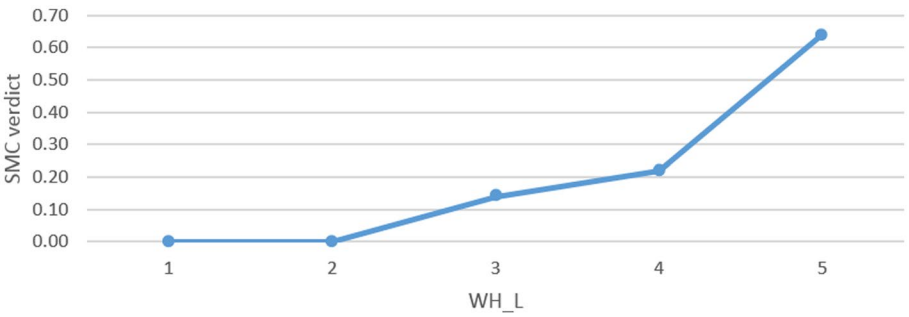

Fig. 8 Probabilities of water height levels on April 27

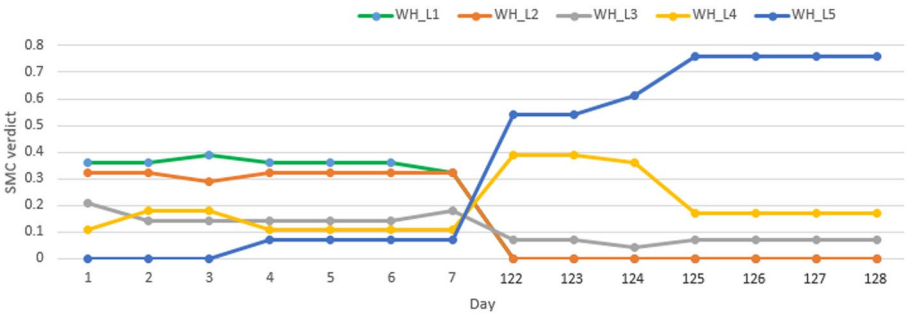

Fig. 9 Probabilities of water height levels at first weeks of January and May

As shown in Fig. 10, there is a high possibility that the water height level will remain at levels 4 or 5 in the last week of May.

Property 4: The probabilities that the water height obtained from the sensor changes from first level (L=1) on January 16th to other levels (L=2..5) on the next day.

In LTL:

$$\begin{cases} P_{=?}[\ (WH\_L = 1\ \&\&\ Day = 16)\ \cup^{3000}\ (WH\_L = L\ \&\&\ Day = 17)\ ]; \\ L = 2 : 5 : 1; \end{cases}$$

Figure 11 shows that change to levels 2 and 3 is most likely while there is little chance of change to levels 4 and 5.

4.2 Qualitative analysis

For qualitative analysis of sensor behavior, we rate sensors' readings based on their probabilities as follows:

1. Not observed (RED): never seen in 28 years.
2. Rare (ORANGE): observed once or twice within 28 years.

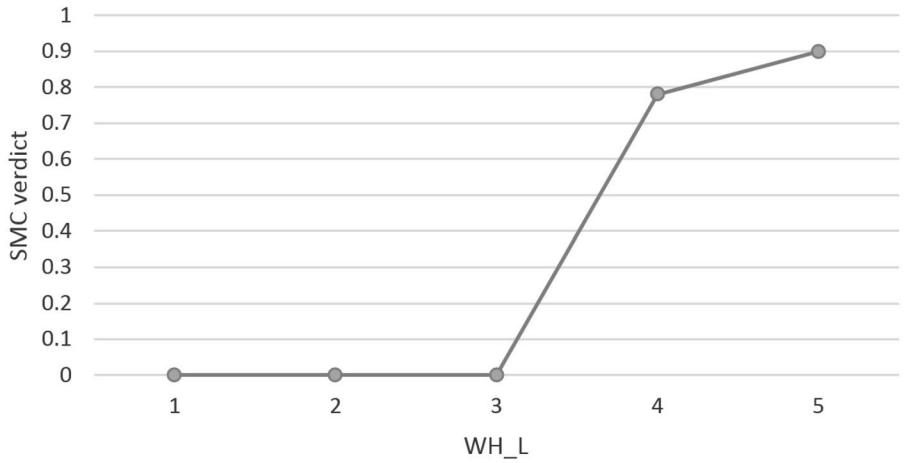

Fig. 10 Results of Property 3

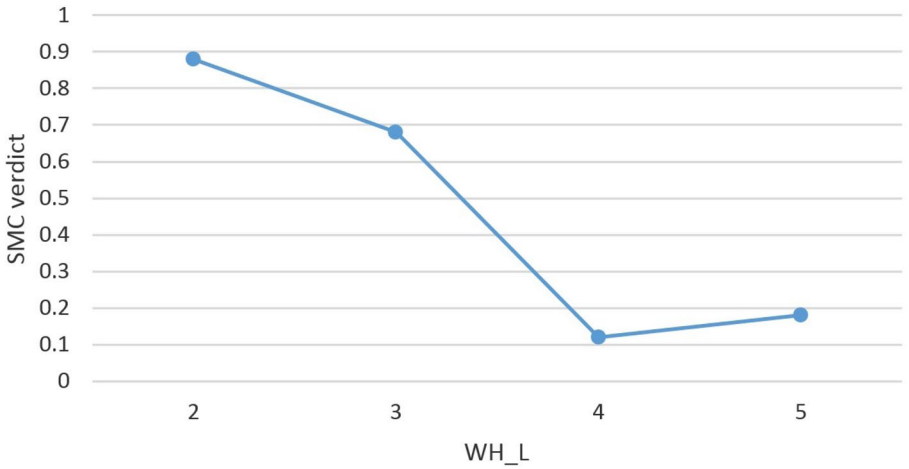

Fig. 11 Results of Property 4

3. Possible (YELLOW): observed 3 to 21 times in 28 years.
4. Very possible (GREEN): observed more than 21 times.

Table 1 defines the possible probabilities. Based on these considerations, we express qualitative properties that allow testing the compliance of sensors' readings with the learned model.

Property 5: Check whether the probabilities that the water height obtained from the sensor reaches level 5 are higher than 0.75.

In LTL: $P_{>0.75}[F^{3000}\ (WH\_L = 5\ \&\&\ Day = T)];\quad T = 1 : 366 : 1;$

Figure 12 shows the results provided by *S*BIP. This property allows calculating the set $DL5_{vp} = \{124, .., 202\}$ of days where the level 5 of water height is very possible.

In the same way, we can calculate the sets $DL4_{vp}$, $DL3_{vp}$, $DL2_{vp}$, $DL1_{vp}$ where levels 4, 3, 2 and 1 are very possible. Based on these calculations, we define the function *isVeryPossibe* as:

$$
\begin{aligned}
&isVeryPossibe(WH\_L, Day) \leftarrow \\
&(WH\_L = 5\ \&\&\ Day \in\ DL5_{vp}\ ||\ WH\_L = 4\ \&\&\ Day \in\ DL4_{vp}|| \\
&WH\_L = 3\ \&\&\ Day \in DL3_{vp}\ ||\ WH\_L = 2\ \&\&\ \ Day \in\ DL2_{vp}\ || \\
&WH\_L = 1\ \&\&\ Day \in\ DL1_{vp})
\end{aligned}
$$

We have also defined the functions *isPossible*, *isRare* and *isNotObserved* which allow, respectively, to check if the data collected by the sensors are possible, rarely observed or never observed. The defined functions are used to build the model of Figure 13 that allows

Table 1 Sensor State Rate

State	Not observed	Rare	Possible	Very Possible
Probability	0	]0, 0.09]	]0.09, 0.75]	]0.75, 1]

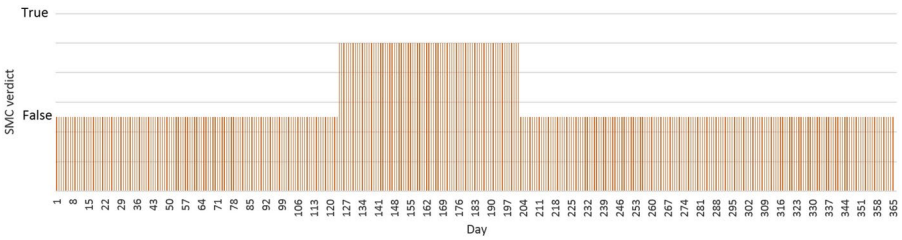

Fig. 12 Probabilities that water height level 5 is very possible

evaluating the conformity or confidence of any water height sensor reading regarding the provided trace. The model can help to distinguish between anomalous and correct sensor readings.

The sensor state model can be used to check the confidence of sensed data from the existing traces. In Fig. 14, we discover very possible readings (Green points), possible readings (Yellow points) and rare readings (Orange points) in the months April and May of 2016. As shown in the figure, some rare readings are detected at the beginning of April and May.

The sensor state model also allows for checking new observations. Figure 15 presents the test results for April and May of 2017. We see that no unusual observation is found and that the observations of Avril are possible and the observations of May are highly possible.

As mentioned in Section 1, our approach could help decision-makers of dam management to evaluate the confidence of sensor readings and identify the possible abnormal

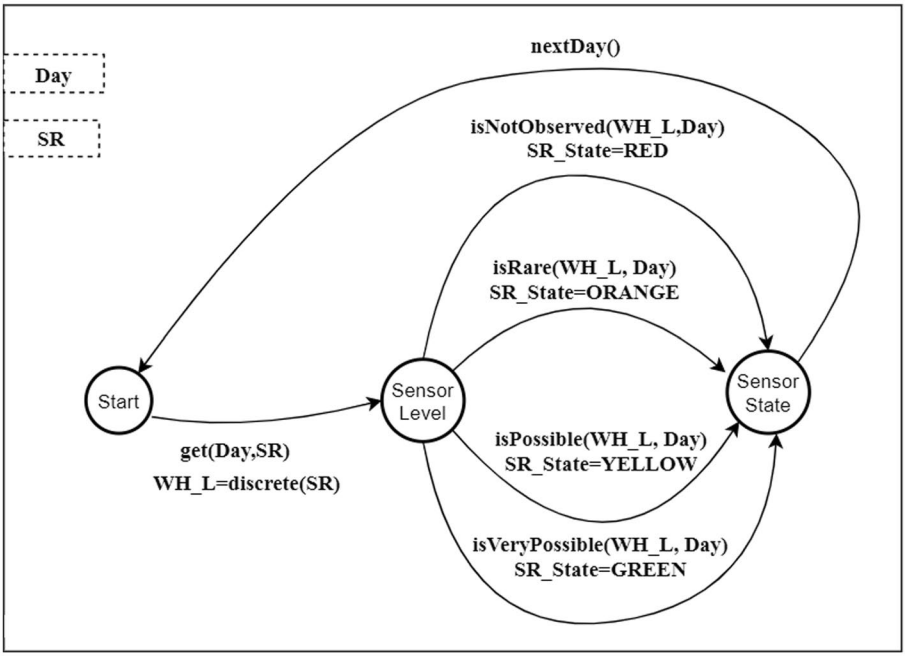

Fig. 13 Sensor State Model

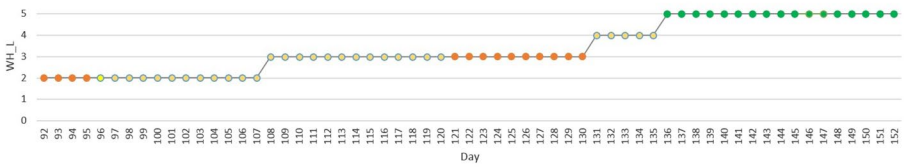

Fig. 14 Score of water height sensor data for April and May of 2016

readings (not observed or rare readings) but not to detect the causes which can be related to sensor malfunction, cyberattacks, change in weather conditions or other reasons. In the future, we plan to enhance our approach by incorporating solutions to confirm the identified abnormal sensor readings and check their causes. We intend to correlate data from a real-time weather reporting system to check if the suspect sensor reading is about changing weather conditions. Other solutions can be considered by combining our approach with IDS (Intrusion Detection Systems) and diagnostic tools for WSN (Wireless Sensor Networks) to check whether the abnormal readings consist of individual sensor faults, hardware malfunction or security-related anomalies. The learned model could be used to detect the cause if it is enriched with failures observation and criteria leading to causes identification.

We note that the sensor behavior models (example of Fig. 7) and the sensor state models (example of Fig. 13) will be updated with the new sensors observations for the new years.

5 Analysis of the collective behavior of sensors

As indicated in the description of our study in Section 1, the spillgate regularization depends on the values of the three sensors which measure *RP*, *WH* and *WF* in the dam infrastructure. So, it is important to have a global forecast on the observations of all the sensors by analyzing their collective behavior.

To analyze the collective behavior of the three sensors from our case study, we need to simulate the execution of all the models specifying their behaviors. As mentioned in Section 2, the BIP framework supports the specification of composite, hierarchically structured components from atomic components. So, we create the BIP compound (composite component) of Fig. 16 that contains the atomic components for *RP*, *WH* and *WF* sensors and the component *Monitor* that scheduler the execution of the sensors' components.

The behaviors of sensors' components are given in Section 3.2. We add external ports to these components in order to interact with the *Monitor* component. The transition from the state $M0$ to the state $M1$ in the *Monitor* component is triggered by the port *PS* that allows starting the execution of the 3 sensors components. The sensors

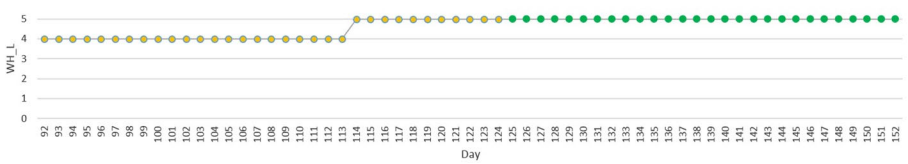

Fig. 15 Score of water height sensor data for April and May of 2017

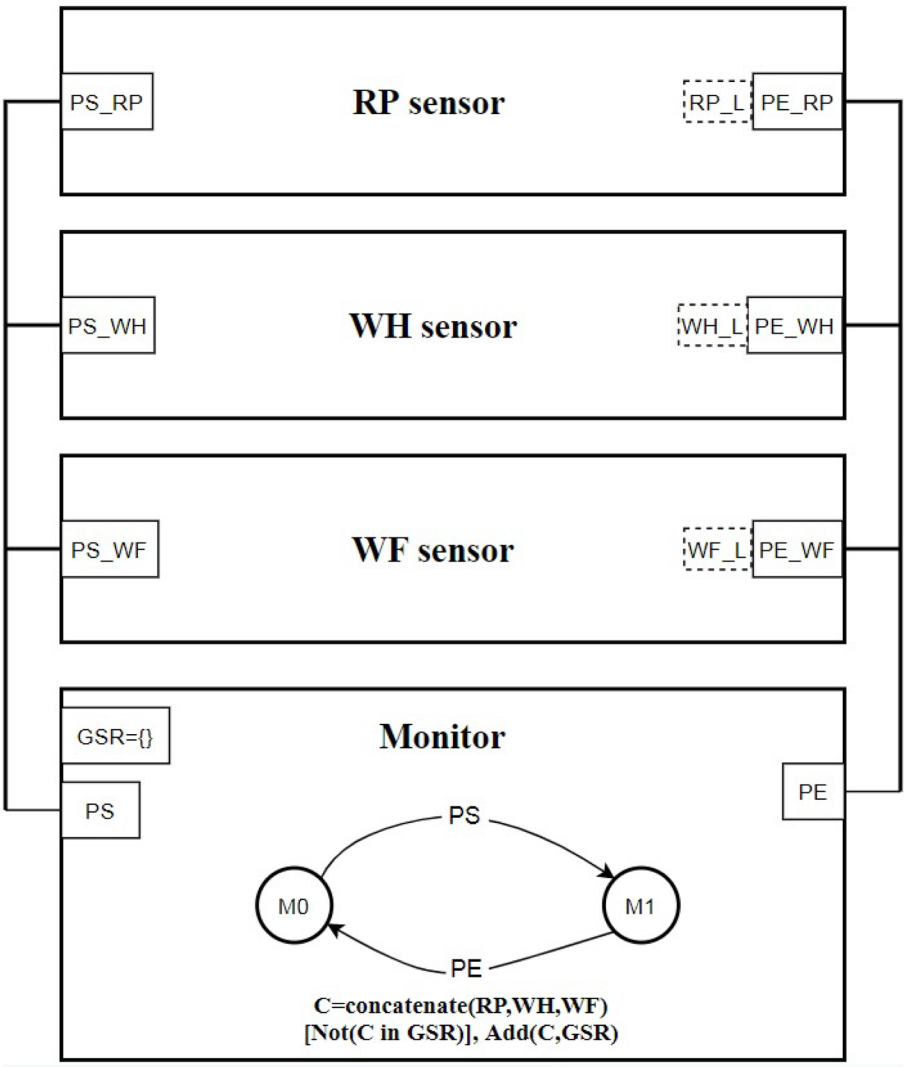

Fig. 16 Architectural Assembly of Sensors Components

components predict the sensors' readings levels (*RP_L*, *WH_L* and *WF_L*) for one day and send these values to the monitor through the ports *PE_RP*, *PE_WH* and *PE_WF*. Synchronous connectors are used for relating these ports with *PE* port of *Monitor* component. After receiving *RP_L*, *WH_L*, and *WF_L* values, we calculate the set *GSR* of the possible groups (classes or configurations) of these values. Each group *C* is a unique combination of *RP_L*, *WH_L*, and *WF_L* represented as a concatenation of these values. *C* represents a class of values obtained by the three sensors that gives a general picture of the water contributions to the dam. Finally, a new iteration will be triggered with the *PS* port for the next day.

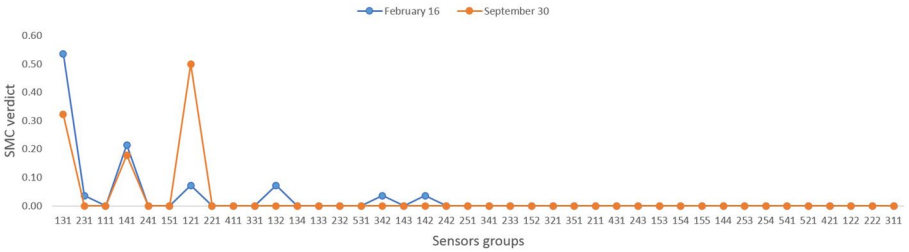

Fig. 17 Results of Property 6

We use *S*BIP to simulate the model of Fig. 16 and to analyze the collective behavior of sensors by expressing quantitative properties to predict sensors configurations for a given period and qualitative properties to test the confidence of a combination of sensors' readings.

5.1 Quantitative analysis

Property 6: The probabilities of obtaining each combination of *RP*, *WH* and *WF* sensors levels on February 16 and September 30.

In LTL:

$\{ P_{=?}[F^{10000}\ (C = gr\ \&\&\ Day = T)];\quad gr \in GSR;\quad T = 47 : 274 : 227;$

Figure 17 presents the results of the property. The abscissa axis shows the forty different possible configurations of sensors levels values. In February 16, the combination $'131'$ ($RP\_L = 1, WH\_L = 3, WF\_L = 1$) is the most likely with probability more than 0.54. The combination $'141'$ is less likely with a probability equal to 0.21. We also see some rare configurations like: $'231', '121', '132', '342', '142'$. From these results, we can conclude that on February 16 the possible values of WH_L are between 2 and 4, and the values of RP_L and WF_L are 1 or 2. On September 30, there are only three possible configurations. The most likely is $'121'$ with a probability of more than 0.5. Configurations $'131'$ and $'141'$ are less likely.

Property 7: The probabilities of obtaining each combination of *RP*, *WH* and *WF* sensors levels, where *RP* is at level 1 and *WH* is at level 5, in the first half of the year.

In LTL: $\begin{cases} P_{=?}[F^{10000}\ (C = gr\ \&\&\ Day = T)];\quad T = 1 : 182 : 1; \\ gr \in \{151, 152, 153, 154, 155\} \end{cases}$

The *S*BIP verdict of property 7 is given in Fig. 18. As shown in Figure, until March 10 the configuration with $RP\_L = 1$ and $WH\_L = 5$ is rarely observed regardless of the WF_L sensor value. From March 10, configuration $'151'$ is possible and even more probable between May 13 and June 30. The other configurations ($'152', '153', '154'$ and $'155'$) remained very rare.

Property 8: The probabilities that each obtained combination of *RP*, *WH*, and *WF* sensors levels keep the same values in the first week of March and the last week of the same month.

In LTL for March 1 to 7:

$\begin{cases} P_{=?}[G^{10000}\ (C = gr\ \&\&\ Day = 61) \cup^{10000}\ (C = gr\ \&\&\ Day = T)]; \\ T = 62 : 67 : 1;\ gr \in GSR \end{cases}$

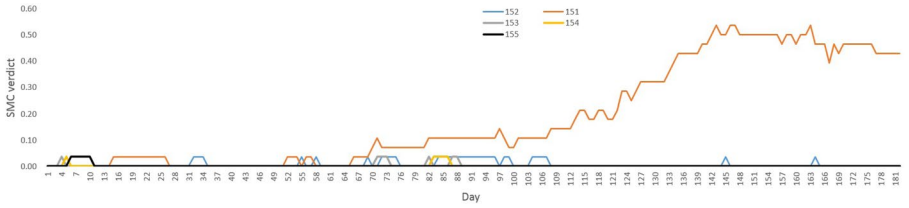

Fig. 18 Results of Property 7

In Fig. 19, we see that there is a high possibility that *RP*, *WH* and *WF* sensors levels stay at configuration ′131′ or ′141′ in both the first and last week of March. In the last week of March, the sensors can also maintain the configurations ′151′ or ′152′.

5.2 Qualitative analysis

In this Section, we express qualitative properties to test whether the probability of a given configuration of *RP*, *WH* and *WF* corresponds to the ranges given in Table 1.

Property 9: check if the probabilities that the obtained *RP*, *WH* and *WF* sensors levels equal 1, 4 and 1, respectively, are greater than 0.75.

In LTL: $P_{>0.75}[F^{10000}\ (C = 141\ \&\&\ Day = T)];\quad T = 1 : 366 : 1;$

In Fig. 20, the results show that property 9 is only true for the set of days $\{115 - 126, .., 206 - 210\}$. We name this set of days where the configuration ′141′ of *RP*, *WH* and *WF* is very possible as $DC141_{vp}$. Other properties have been defined to find the sets of days where the several configurations shown in Fig. 17 are very possible.

As in Section 4.2, we define the following functions based on the results of the qualitative properties:

- *isVeryPossibeConfiguration*(*C*, *Day*)
- *isPossibleConfiguration*(*C*, *Day*)
- *isRareConfiguration*(*C*, *Day*)
- *isNotObservedConfiguration*(*C*, *Day*)

These functions that check if a combination of *RP*, *WH*, and *WF* levels is *very possible*, *possible*, *rarely observed*, or *never observed* are used to specify a model (*Configuration State Model*) with the same pattern as that of Fig. 13 (*Sensor State Model*). We use this

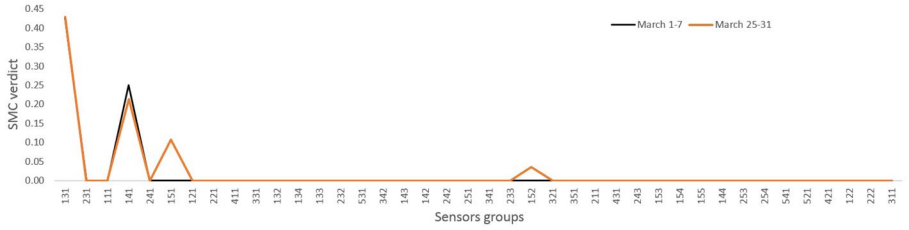

Fig. 19 Results of Property 8

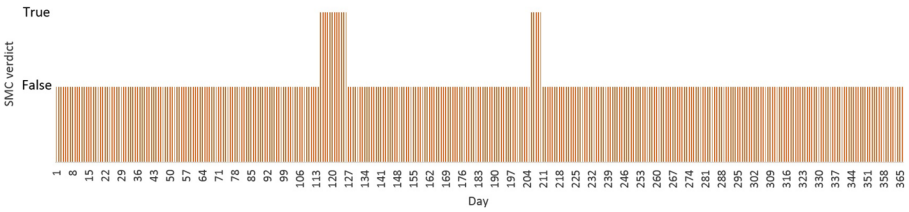

Fig. 20 Results of Property 9

new model to test the collective behavior of sensors over time. The results for November and December of 2017 are given in Fig. 21.

As shown in Figure, most of the configurations observed are possible (Yellow points). There are some very possible configurations at the end of December (Green points). The results also show that two configurations are never seen before according to the sensors traces (Red points). These are configuration ′211′ recorded on November 24, 2017, and configuration ′411′ recorded on December 10, 2017.

We sought explanations on the red points in the data provided by our industrial partner. After checking the weather readings in these days, we found that these are due to exceptional levels of rain precipitation not recorded in previous years on November 24 and December 10. As mentioned in Section 4.2, we plan to improve our approach by correlating data from a real-time weather reporting system to eliminate these false positives caused by changing weather conditions.

6 Related work

Time series analysis is one of the active areas of research due to its application in different fields, such as in the context of IoT-based systems. For time series data from sensors, predicting the next measurements and detecting erroneous readings are the relevant tasks. (Giannoni et al., 2018) presents the several approaches proposed for this purpose:

– Statistical approaches such as the method proposed by (Yu et al., 2014) that builds a window-based forecasting model from past observations, then it classifies the sensors' readings as anomalous based on a given prediction confidence interval.

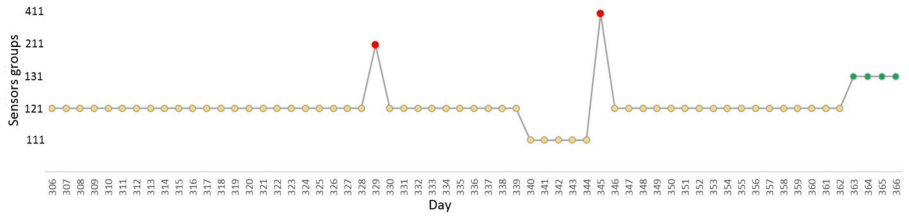

Fig. 21 Score of dam sensors configurations for November and December of 2017

- Probabilistic approaches use probabilistic models such as Bayesian Networks (BNs) (Hill et al., 2009) to measure the probability of sensors' readings. However, these approaches do not scale well.
- Proximity-based or clustering-based approaches such as (Breunig et al., 2000; He et al., 2003) use distances between the sensed data to detect the erroneous readings. For high-dimensional data, these approaches do not work well.
- Prediction-based approaches such as (Malhotra et al., 2015; Shahid et al., 2015) use machine learning methods to predict the sensors' readings based on a model trained from past observations. However, training is time-intensive.

In this paper, we propose a new approach that allows building a component-based data-driven models for predicting sensors' readings and evaluating their conformity with past observations. The models are built for each type of sensor independently (in isolation) from existing traces, so the number of sensors does not influence scalability. The length of traces impacts the accuracy of the model but has limited impact on scalability. Indeed, our approach is different from all the approaches presented above. It allows to build a behavioral automata-based model from data and analyze this model using formal verification techniques. Among the works in this direction:

- The authors in (Saives et al., 2015) use Extended Finite Automata and residuals techniques to detect deviations of the behavior of the inhabitant in a smart home from a log of binary sensor events.
- (Mercaldo et al., 2019) models logs from SCADA systems using timed automata and applies the UPPAAL model checker to express a set of logic properties for detecting attacks targeting these systems.
- (Naskos et al., 2016) uses Markov Decision Process for modeling the behavior of elastic cloud applications based on past log and then introduces probabilistic model checking to perform cloud elasticity decision using PCTL.
- (Franco et al., 2016) specifies a stochastic model in Deterministic-Time Markov Chain from the architecture description of the managed system considering different metrics related to cloud-infrastructure execution traces. Then, the PRISM model checker is used to optimize the self-adaptation decisions.

In our approach, we generate stochastic automata expressed by the BIP that specify the sensors' behavior based on sensors traces. Then, we use the *S*BIP to simulate the learned models and express LTL properties that predict the sensors' readings and analyze the individual and collective behavior of sensors in time. In the above, we listed the most similar-related approaches, and we believe that the sensors models could have been obtained and analyzed using some of these formalisms and their associated tools, with comparable effort and performance. Among the specificities of our approach :(i) it follows a component-based approach supported by the BIP framework that facilitates integrating stochastic behaviors, and also the reutilization and maintenance of components, (ii) Statistical Model Checking (SMC) using *S*BIP relies on simulation-based techniques known to be less memory intensive than standard model-checking or probabilistic model-checking techniques. Using SMC, executions are first sampled, after which statistical techniques are applied to determine whether a given property holds. SMC techniques have been applied for the analysis of various case studies such as autonomous driving controllers (Barbier et al., 2019) and biological systems (David et al., 2015b).

Several frameworks exist for modeling and analyzing stochastic systems, especially, statistical model checking has drawn lot of interest in the research community as UPPAAL-SMC (David et al., 2015a), PRISM-SMC (Kwiatkowska et al., 2011), MRMC (MRMC, 2011), Ymer (Younes, 2005) and (COSMOS, 2015). For instance, PRISM implements SMC techniques such as Probability Estimation techniques (PE) and Hypothesis Testing, and the model to be checked is constructed before and stored in memory. MRMC offers SMC with confidence interval computation. However, it always loads Markov chain representations into memory completely. Ymer considers Generalized Semi Markov Processes (GSMP) and Continuous Time Markov Chains (CTMC) using the PRISM dialect and uses a numeric-symbolic engine from PRISM. COSMOS uses confidence interval computation and exhibits performance comparable to PRISM on several benchmarks (Ballarini et al., 2015). UPPAAL-SMC and Ymer are closer to *S*BIP, and both of them consider GSMP. Comparing *S*BIP to UPPAAL-SMC and other SMC tools, *S*BIP supports a component-based language (BIP) endowed with capabilities to express automata-based and/or Petri Net behavior. *S*BIP is also more powerful since it has more capabilities relayed to the behavior description while constructs are based on *C++*. Moreover, checking the model relies on processing the traces resulted from the model execution. *S*BIP was applied for the analysis of various systems (Beaulaton et al., 2019; Nouri et al., 2015; Nouri et al., 2018). For a deeper understanding of the SMC tools, we refer to the survey in (Agha & Palmskog, 2018).

7 Conclusion

We presented a component-based approach for formal modeling and analysis of sensors' behavior. A formal model expressed as stochastic automata has been derived from sensor time series data then quantitative LTL properties expressed on this model are used to predict sensor readings. Also, qualitative LTL properties are used for defining an automata-based model that allows checking if the new measurements are compliant with past observations. We have applied our approach to analyzing the behavior of three sensors from a dam infrastructure at different times. In our approach, we have also assembled the behavior models of different sensors to observe their collective behavior over time which helps to control the spillgate and detect any inconsistencies between sensors' behaviors.

Our approach provides several advantages, including:

- We use BIP formalisms that allow the rigorous specification and analysis of sensors' behavior.
- We use a component-based approach supported by BIP that facilitates portraying sensors behavior with reusability, and maintainability features.
- We developed a prototype that automatically generates sensor behavior and sensor state models from any existing traces.
- We use statistical model checking that relies on stochastic simulation and therefore is not impacted by the size of the state space of the model.

In the future, we will work on the integration of machine learning algorithms such as K-Nearest-Neighbors (K-NN) for the analysis of sensors' behavior. We plan to use K-NN to group the data into a set of classes and learn information about timed switching from one class to others. Then, we can use SMC to check the changes of the sensor data and

identify the reason for this variation regarding other sensed data. We also plan to implement the solutions discussed in Section 4.2 for detecting the causes of the abnormal behavior of sensors.

Acknowledgements The authors would like to thank EMALCSA Company for the data collected from the dam infrastructure.

Funding Information The research leading to these results has been supported by the European Union through the BRAIN-IoT project H2020-EU.2.1.1. Grant agreement ID: 780089.

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies involving animals or human participants performed by any of the authors

References

Agha, G., & Palmskog, K. (2018). A Survey of Statistical Model Checking. *ACM Transactions on Modeling and Computer Simulation, 28*(1), 1–39. https://doi.org/10.1145/3158668

Al-Turjman, F., & Malekloo, A. (2019). Smart parking in IoT-enabled cities: A survey. *Sustainable Cities and Society, 49,* 101608.

Alur, R., & Henzinger, T. (1993). Real-Time Logics: Complexity and Expressiveness. *Information and Computation, 104*(1), 35–77. https://doi.org/10.1006/inco.1993.1025

Alvarez Carmona, M. A., Carrasco Ochoa, J. A., & Martinez Trinidad, J. F. (2013). Combining techniques to find the number of bins for discretization. In: 2013 32nd International Conference of the Chilean Computer Science Society (SCCC), pp 54–57. https://doi.org/10.1109/SCCC.2013.11

Ballarini, P., Barbot, B., Duflot, M., Haddad, S., & Pekergin, N. (2015). Hasl: A new approach for performance evaluation and model checking from concepts to experimentation. *Performance Evaluation, 90,* 53–77.

Barbier, M., Renzaglia, A., Quilbeuf, J., Rummelhard, L., Paigwar, A., Laugier, C., Legay, A., Ibanez-Guzman, J., & Simonin, O. (2019). Validation of perception and decision-making systems for autonomous driving via statistical model checking. *In: 2019 IEEE Intelligent Vehicles Symposium (IV)*, pp 252–259.

Basu, A., Bensalem, S., Bozga, M., Combaz, J., Jaber, M., Nguyen, T. H., & Sifakis, J. (2011). Rigorous Component-Based System Design Using the BIP Framework. *IEEE Software, 28*(3), 41–48.

Beaulaton, D., Said, N. B., Cristescu, I., & Sadou, S. (2019). Security Analysis of IoT Systems Using Attack Trees. In M. Albanese, R. Horne, & C. W. Probst (Eds.), *Graphical Models for Security* (Vol. 11720, pp. 68–94). Cham: Springer International Publishing.

Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000). LOF: identifying density-based local outliers. *ACM SIGMOD Record, 29*(2), 93–104. https://doi.org/10.1145/335191.335388

Chehida, S., Baouya, A., Bensalem, S., & Bozga, M. (2020). Applied statistical model checking for a sensor behavior analysis. In R. Pérez-Castillo (Ed.), *Shepperd M, Brito e Abreu F, Rodrigues da Silva A* (pp. 399–411). Springer International Publishing, Cham: Quality of Information and Communications Technology.

COSMOS. (2015). *Cosmos tool*. http://www.lsv.ens-cachan.fr/Software/cosmos/

Daissaoui, A., Boulmakoul, A., Karim, L., & Lbath, A. (2020). IoT and Big Data Analytics for Smart Buildings: A Survey. *Procedia Computer Science, 170,* 161–168. https://doi.org/10.1016/j.procs.2020.03.021

David, A., Larsen, K. G., Legay, A., Mikučionis, M., & Poulsen, D. B. (2015a). Uppaal SMC tutorial. *International Journal on Software Tools for Technology Transfer 17*(4), 397–415. https://doi.org/10.1007/s10009-014-0361-y

David, A., Larsen, K. G., Legay, A., Mikucionis, M., Poulsen, D. B., & Sedwards, S. (2015b). Statistical model checking for biological systems. *International Journal on Software Tools for Technology Transfer, 17*(3), 351–367. https://doi.org/10.1007/s10009-014-0323-4

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In: Prieditis A, Russell S (eds) *Machine Learning Proceedings* 1995, Morgan Kaufmann, San Francisco (CA), pp 194 – 202. https://doi.org/10.1016/B978-1-55860-377-6.50032-3

Franco, J. M., Correia, F., Barbosa, R., Zenha-Rela, M., Schmerl, B., & Garlan, D. (2016). Improving self-adaptation planning through software architecture-based stochastic modeling. *Journal of Systems and Software, 115,* 42–60. https://doi.org/10.1016/j.jss.2016.01.026

Giannoni, F., Mancini, M., & Marinelli, F. (2018). *Anomaly Detection Models for IoT Time Series Data*. https://arxiv.org/abs/1812.00890

He, Z., Xu, X., & Deng, S. (2003). Discovering cluster-based local outliers. *Pattern Recognition Letters, 24*(9–10), 1641–1650. https://doi.org/10.1016/S0167-8655(03)00003-5

Hérault, T., Lassaigne, R., Magniette, F., & Peyronnet, S. (2004). Approximate probabilistic model checking. *Verification, Model Checking, and Abstract Interpretation* (pp. 73–84). Berlin Heidelberg, Berlin, Heidelberg: Springer.

Hill, D. J, Minsker, B. S., & Amir, E. (2009). Real-time Bayesian anomaly detection in streaming environmental data: Real-time bayesia anomaly detection. *Water Resources Research 45*(4). https://doi.org/10.1029/2008WR006956

Kwiatkowska, M., Norman, G., & Parker, D. (2011). Prism 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan & S. Qadeer (Eds.), *Computer Aided Verification* (pp. 585–591). Heidelberg: Springer, Berlin Heidelberg, Berlin.

Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). Long short term memory networks for anomaly detection in time series. In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges, Belgium.

Mediouni, B. L., Nouri, A., Bozga, M., Dellabani, M., Legay, A., & Bensalem, S. (2018). SBIP 2.0: Statistical Model Checking Stochastic Real-time Systems. *In: ATVA 2018 - 16th International Symposium Automated Technology for Verification and Analysis*, Springer, Los Angeles, CA, United States, pp 536–542. https://doi.org/10.1007/978-3-030-01090-4_33

Mercaldo, F., Martinelli, F., & Santone, A. (2019). Real-Time SCADA Attack Detection by Means of Formal Methods. *In: 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, IEEE, Napoli, Italy, pp 231–236. https://doi.org/10.1109/WETICE.2019.00057

MRMC. (2011). Mrmc tool. http://www.mrmc-tool.org

Naskos, A., Gounaris, A., Mouratidis, H., & Katsaros, P. (2016). Online Analysis of Security Risks in Elastic Cloud Applications. *IEEE Cloud Computing, 3*(5), 26–33. https://doi.org/10.1109/MCC.2016.108

Nouri, A., Bensalem, S., Bozga, M., Delahaye, B., Jegourel, C., & Legay, A. (2015). Statistical model checking QoS properties of systems with SBIP. *International Journal on Software Tools for Technology Transfer, 17*(2), 171–185.

Nouri, A., Mediouni, B. L., Bozga, M., Combaz, J., Bensalem, S., & Legay, A. (2018). Performance evaluation of stochastic real-time systems with the SBIP framework. *International Journal of Critical Computer-Based Systems, 8*(3/4), 340.

Park, C., Kim, Y., & Jeong, M. (2018). Influencing factors on risk perception of IoT-based home energy management services. *Telematics and Informatics, 35*(8), 2355–2365.

Pnueli, A. (1977). The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science, *IEEE Computer Society*, USA, pp 46–57. https://doi.org/10.1109/SFCS.1977.32

Saives, J., Pianon, C., & Faraut, G. (2015). Activity Discovery and Detection of Behavioral Deviations of an Inhabitant From Binary Sensors. *IEEE Transactions on Automation Science and Engineering, 12*(4), 1211–1224. https://doi.org/10.1109/TASE.2015.2471842

Shahid, N., Naqvi, I. H., & Qaisar, S. B. (2015). One-class support vector machines: analysis of outlier detection for wireless sensor networks in harsh environments. *Artificial Intelligence Review, 43*(4), 515–563. https://doi.org/10.1007/s10462-013-9395-x

Stewart, W. J. (2009). *Probability, Markov chains, queues, and simulation: the mathematical basis of performance modeling*. Princeton University Press.

Tao, Z. (2020). Advanced Wavelet Sampling Algorithm for IoT based environmental monitoring and management. *Computer Communications, 150,* 547–555. https://doi.org/10.1016/j.comcom.2019.12.006

Yang, Y., Webb, G. I., & Wu, X. (2010). *Discretization Methods* (pp. 101–116). US, Boston, MA: Springer.

Xie, Yi., & Shun-Zheng, Yu. (2009). A Large-Scale Hidden Semi-Markov Model for Anomaly Detection on User Browsing Behaviors. *IEEE/ACM Transactions on Networking, 17*(1), 54–65. https://doi.org/10.1109/TNET.2008.923716

Younes, H. L. S. (2005). Ymer: A statistical model checker. *Computer Aided Verification* (pp. 429–433). Berlin Heidelberg: Springer.
Younes, H. L. S., & Simmons, R. G. (2002). Probabilistic verification of discrete event systems using acceptance sampling. In E. Brinksma & K. G. Larsen (Eds.), *Computer Aided Verification* (pp. 223–235). Berlin Heidelberg: Springer.
Yu, Y., Zhu, Y., Li, S., & Wan, D. (2014). Time Series Outlier Detection Based on Sliding Window Prediction. *Mathematical Problems in Engineering, 2014,* 1–14. https://doi.org/10.1155/2014/879736

Salim Chehida is a PhD in computer science since 2017 from University of Oran 1 in Algeria in collaboration with University of Grenoble Alpes in France. Since March 2018, Salim Chehida is a researcher at University of Grenoble Alpes (LIG laboratory, VERIMAG laboratory). His research interests include Software Engineering, Verification and Validation of Software Systems, Semi-formal and Formal Specification Methods (UML, B, etc.), Model Driven Engineering, Runtime Verification, Information Systems Security, IoT, and CPS.

Abdelhakim Baouya is a Ph.D. in Computer engineering. Currently, he is a researcher at Verimag laboratory, University of Grenoble Alpes. He is working on Software Architecture specification (UML/MARTE, SysML, AADL, Autofocus AF3), Formal verification and, Code generation. He is interested in developing formal methods, techniques, and tools for IoT and Cyber-Physical Systems. Some of his interests are Model-based Design, Dependability Analysis, Stochastic Component-based Design, Statistical/Probabilistic Model-checking.

Saddek Bensalem is a Full Professor at the University of Grenoble Alpes, France. He received his PhD in Computer Science from the INP Grenoble. His area of expertise is modelling and validation of real-time systems.

Marius Bozga graduated the Faculty of Mathematics and Computer Science, Babes-Bolyai University of Cluj-Napoca, Romania in 1995. He received his Ph.D. in Computer Science from the Joseph Fourier University of Grenoble in 1999. Since 2000, Marius Bozga is CNRS research engineer and member of the VERIMAG laboratory in Grenoble. His research interests are focused on component-based design for distributed real-time systems and include formal models for components, model-based design and implementation, automatic validation methods and tools.