A Hierarchical Automata Based Approach for Anomaly Detection in Smart Home Devices

Kai Kang[1,2], Lijie Xu[1,2,3], Wei Wang[1,2,3], Guoquan Wu[1,2,3], Jun Wei[1,2,3], Wei Shi[4], Jizhong Li[4]
[1]State Key Lab of Computer Science, Institute of Software, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]Institute of Software Technology, Chinese Academy of Sciences, Nanjing
[4]Huawei

***Abstract*—Smart home is an important application scenario of the Internet of Things. However, smart home tends to suffer from runtime failures due to its complex software/hardware, unexpected/wrong human operations, environmental changes, etc. It is hard to detect the device anomalies, since smart devices have various types, various failure types, and they also generally do not expose the inner runtime information. Current anomaly detection techniques for general programs or distributed systems do not apply for smart home devices.**

To help both device manufactures and users to detect the device anomalies, we propose a general framework named SmartHomeDetector. It first builds a hierarchical automata based behavior model during the testing phase, and uses it as the baseline of IoT device. At runtime, a comprehensive anomaly detection technique is proposed to identify various device anomalies in time. We implement SmartHomeDetector based on the open source smart home system HomeAssistant. Our experiments and case studies on real-world smart home devices show that SmartHomeDetector is effective. It accurately models the running behavior of IoT devices and can detect various device anomalies at runtime.

***Keywords*—Smart Home, Hierarchical Automata, Fault Detection**

I. Introduction

The Internet of Things (IoT) is a well-established paradigm [11] nowadays. It is fueling remarkable innovation in which ordinary physical objects in our environment are equipped with Internet connectivity, sensing, control, and computing capabilities. Recent years have witnessed the rapid progress of IoT technologies, with many of them already seeing wide adoption. Examples include smart plugs, smart door locks, smart bulbs and many others. People, in fact, interact daily with a growing number of Internet-enabled devices in many different contexts, ranging from their smart homes to smart cities. According to a recent report [1], the number of IoT devices is projected to reach 20.4 billion in 2020, forming a global market valued $3 trillion.

A smart home refers to a regular home augmented with various types of IoT devices that provide services (i.e, IoT services) for improving comfort and convenience of occupants' daily lives. In existing smart home scenarios, a control center, such as mobile phone, smart speaker, and smart TV, usually controls smart home devices within a certain range. Platforms such as HomeKit [4], Google Assistant [2] and Home Assistant [3], provide a good foundation for smart home device discovery, connection and deployment.

Because IoT devices are vastly heterogeneous and execute in a wide range of remote locations and operating conditions, they are subject to frequent hardware and software failures and performance degradations. The abnormal situation of the smart home equipment may occur suddenly or cumulatively. It may be caused by human operation or non-human factors such as long running, sudden environmental changes, or aging of some component in the device. However, most existing common smart home control platforms only provide some simple viewing and control capabilities. If the anomaly cannot be detected and handled in time, it will break the functionality of some components in the device, leading to the corruption of the device in the end. Furthermore, many of failures in the device may corrupt the data it produces, and affect the "downstream" computation that depends on this data (e.g., the action in the trigger-action rules).

To help both device manufactures and users to detect various device anomalies at runtime, this paper proposes SmartHomeDetector, a hierarchical automata based approach for anomaly detection in smart home devices, which consists of two phases: model construction and runtime anomaly detection. In the phase of model construction, a random walking based testing approach is adopted to construct a hierarchical automata based behaviour model, which not only captures external user operations, but also inner device transformations, and various runtime metrics, to represent the baseline of the device under test.

When the device is deployed into the smart home environment, our approach enters into the phase of anomaly detection. Based on constructed baseline model, a comprehensive anomaly detection technique is proposed, which tracks the running behaviour of the device, and checks it with baseline model to identify four types of deviations caused by abrupt attribute change, accumulated change, component failure and human/environment factors.

We implement SmartHomeTracer based on the open source smart home system HomeAssistant. Our experiments and case studies on real-world smart home devices show that SmartHomeTracer is effective and it can accurately detect various types of device anomalies.

Generally, this paper makes the follow contributions:

- We propose SmartHomeDetector, a novel hierarchical automata based approach to detect anomalies in IoT devices, which first builds a baseline model of IoT device using automatic testing approach, and then designs a comprehensive detection technique to identify various anomalies in IoT devices at runtime.
- An implementation of our SmartHomeDetector approach, and a thorough empirical evaluation whose results show that proposed technique is effective, and it can accurately detect various anomalies in IoT devices.

II. Background

A. Smart home devices

Smart home devices are a branch of smart devices, mainly focusing on devices in the home environment. By analyzing common smart home devices, smart home device information can be divided into *modes*, *operations* and *attributes*. Modes indicate the running stages of devices, such as the high-speed mode of the air purifier and the low-power mode of the humidifier. Operations refer to the mode conversion conditions of devices. For example, users can change the wind speed from high to low, and the device can automatically adjust to the low power mode according to the environmental information. Attributes are the device's runtime indicators, such as the AQI(air quality index) displayed by the air purifier and the humidity value displayed by the humidifier.

The above information is very common for smart home devices, and it is also important factors to be considered when building models. Next we analyze the characteristics of smart home devices, which can be divided into static characteristics and dynamic characteristics according to whether the devices are running. Specifically, smart home devices have the following static characteristics: (1) There are many types of devices. (2) Device's attributes are different from each other. For example, some devices can obtain environmental information (such as air quality) and adapt to changes in the environment, but some cannot perceive changes in environmental information. (3) The equipment can be interconnected. For example, purifiers using humidifier water mist as pollutant particles [5]. (4) Fast device replacement.

The dynamic characteristics of smart home devices are summarized as follows: (1) There are multiple modes of devices. (2) The attributes value of the device are related to the mode. (3) The operations that can be performed on the device are complex. (4) Attributes may be affected by time. For example, the device temperature may gradually increase with time accumulation.

The various and variable characteristics of smart home devices increase the difficulties when modeling smart home devices. In the modeling process, it is necessary to build a general model for smart home devices, which can cover the characteristics of the devices and make a reasonable record and display of its running status.

B. Home Assistant

The Home Assistant [3] of the open source community is a mature and complete smart home control platform that supports highly customized settings such as automation, grouping, and UI customization. Users can use Home Assistant platform to automatically link external devices according to their own needs.

One can manually call services registered by the device through Home Assistant front-end page, or build some automation components through configuration information. The instructions issued by the user and the automation component must first pass through Home Assistant core, which is an information relay station between the user and the device entity. Home Assistant core does not directly interact with the external environment, but connects to the external world through components. Each component is responsible for a specific domain in Home Assistant. Components can interact with the outside world, such as obtaining exchange rates, performing light-on actions, and reading temperatures from temperature sensors. Home Assistant supports component extension, and users can easily add smart home devices to Home Assistant when needed.

III. Hierarchical automata model

In this section, we first define our hierarchical automata based behaviour model and then describe how to build this model via automatic testing approach.

A. Model definition

In Section II, we have summarized the static and dynamic information of smart home devices into three elements: modes, operations, and attributes. Our behaviour model combine these three elements into a hierarchical automata model, which consists of the following parts:

(1) State. States are used to describe the running status of smart home devices. As shown in Fig 1, we divide states into internal states and external states. External states indicate modes of devices, such as cooling mode and sleep mode for air conditioning device. Each external state has multiple responding internal states, which indicate multiple device attributes recorded at a time point. For example, the temperature is 36 degrees at 12 o'clock and 37 degrees at 12:30.

(2) Edge. Edges record the transition relationship between states. For external states, external edges record human operations and the device's automatic conversion operations. For internal states, internal edges record time information.

(3) Diagram. The external state transition diagram records the mode set of the device and the conversion relationship between different modes. The internal state transition diagram records the values of various attributes of the device that change with time in the same mode.

The entire model is divided into multiple modules according to modes, and the modules are converted by operations. Each module includes corresponding attributes, which are represented in a fixed vector form. At the same time, each module is further divided with time as the standard. This model

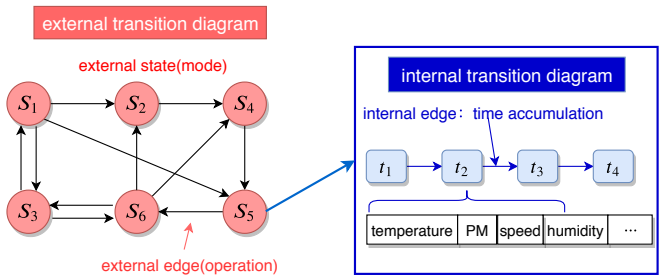

Fig. 1. Hierarchical automata model.

representation method for multi-level division and state transition is similar to the hierarchical state model. We improve the existing hierarchical state model to make it more suitable for the characteristics of smart home devices.

B. Modeling process

The hierarchical automata model can be divided into information model, structural model, and behaviour model according to its completeness. Next, we will introduce the construction process of the hierarchical state model of the smart home device step by step.

Step 1: obtain the mode set, operation set and attribute set of the device to build information model. We can use Home Assistant to get the above information of the device by sending a request to the device through WebSocket. For example, we can send "*get_states*" request to obtain a device's mode set and attribute set, and send "*get_services*" request to obtain a device's operation set.

Step 2: filter operations and attributes to build structural model. Since the external state in the model corresponds to the operation mode of the device, the selection criterion for the current operation is whether the operation can change the mode of the device. Specifically, we select the elements in the operation set one by one, and then perform these operations on the device separately. Next, we obtain the mode before and after the operation and compare whether the two are the same. If they are different, it means that the current operation can change the mode. Add the operation to the final operation set and proceed to the filtering process of the next operation. Otherwise the current operation is discarded. Attribute filtering method is to discard text attributes and boolean attributes, and select the numeric attributes as the attribute set, because text attributes and boolean attributes are less related to modes and attributes of devices and do not change much, which are difficult to analyze.

Step 3: according to the user's behaviour path, combine structured information to build a behaviour model. Now we get the vector representation of external states, external edges, and internal states in the hierarchical state automaton model. But to form a complete hierarchical state automaton model, we also need to connect vertices, edges and vectors. We propose a test method based on random walk. Figure 2 shows the process of the test method based on random walk, which involves model, devices, test tools and generated data.

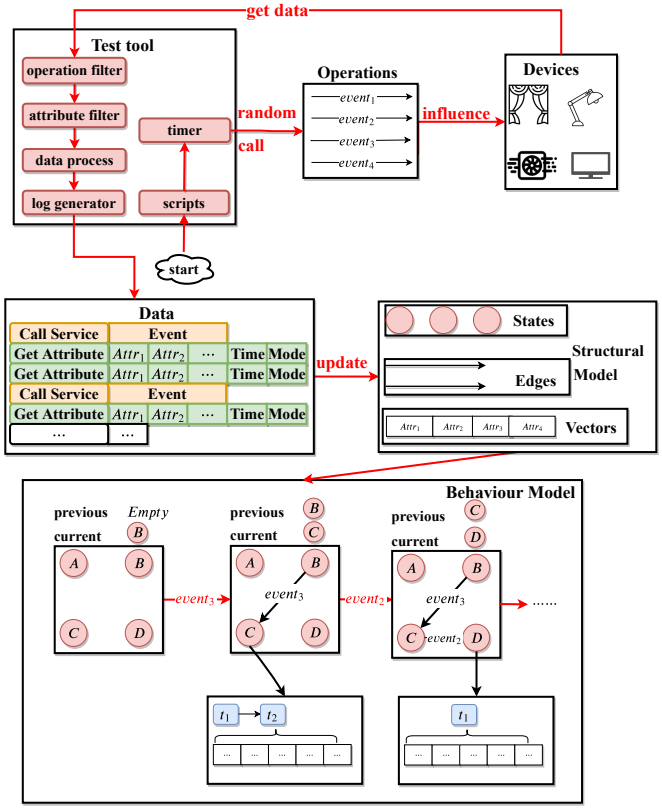

Fig. 2. Test method based on random walk.

The test tool is used to regularly obtain real-time data of the device and generate log information on the data. Each line in the log file represents a piece of data. "Call Service" represents the operation data, and the data information is the operation name. "Get Attribute" represents the attribute data, and the data information is the filtered attribute value of each dimension, the current mode and time. When processing operation data, we need to record the modes before and after the device performs the operation, and add an external edge between the two operation modes. When processing attribute data, the new internal state is added in absolute time in the internal state transition diagram, or the internal state in the same time interval is updated in relative time. The processing of operational data can associate different external states and external edges. The processing of attribute data can associate attributes, time and states to build a complete behaviour model.

Through the above process, a behaviour model can be constructed according to the simulated operation. However, the random operation call makes the traversal order of the external state not fixed, resulting in the test process may not completely cover the model for a long time. Therefore, it is necessary to introduce some termination methods of the modeling process to judge whether the model converges. The modeling process includes the following three termination methods:

(1) Fixed data amount: this method limits the amount of data used in the modeling process. The specific data amount is determined by the type of devices and user parameters. And

different amounts of data will affect the accuracy and time-consuming of modeling.

(2) Sliding window: this method specifies a sliding window of length T, in which the latest updated T state automaton models are recorded. Whenever a new piece of data arrives, the updated model of the current data is added to the window first. Then determine whether the first state in the window is exactly the same as the last state. If they are the same, it means that the external state of the model has not been updated after T pieces of data are updated. If it is different, slide the window backward by one model position, and continue to process the next data until the model converges.

(3) Enumerate: each external state saves a set of operations. The initial value is the complete set of filtered operation sets. Each set of external states are independent of each other. Each time the operation is randomly called, the element is selected from the current set of external states, and then the element is deleted from the set, so that when the set of each external state is empty, the model is considered to converge and the modeling process is ended.

If the random call paths are similar, the exhaustive method has the highest accuracy, the fixed data amount method has the lowest, and the convergence speed is the opposite. The sliding window method is all at a medium level. If the call path is very different, the results are likely to be completely different.

IV. Anomaly Detection

Based on our hierarchical state automata model, we design several approaches for detecting four main types of anomalies.

A. Abrupt anomaly

Abrupt anomaly refers to the sudden change of attribute value of the device, which is specifically manifested as a certain attribute value exceeds the previous recording threshold. The hierarchical state automaton model can judge whether the attribute value has abrupt change according to the mode of the current attributes data. The mode boundary can be defined as the range boundary corresponding to each attribute value in the external state. And different range boundary determination methods are adopted for the attributes of different change trends. As shown in Figure 3, for the attributes of the fluctuation trend, the attribute value range fluctuates within a range, so the maximum and minimum values of the attribute value can be recorded. For the attribute of the linear change trend, its value is closely related to time, so the values of each time point are recorded. When there are errors, some value ranges– the range boundary of the attribute of the fluctuation trend–can be increased proportionally. After the range boundaries of all attribute sets are obtained, the mode boundaries of the current operation mode are formed. If one or more attributes of real-time data are beyond the mode boundary, the corresponding attribute is considered to be abnormal; otherwise, the current attribute data is considered to be in a normal state.

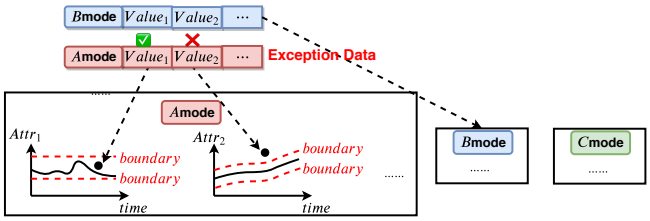

Fig. 3. Mode boundary.

B. Accumulated anomaly

Accumulated anomaly is that during the runtime of the device, the attribute value changes within the normal range, but the overall change trend of the attribute value is not consistent with the normal situation. The change trend includes the average level and fluctuation of the attribute value. The hierarchical state automaton model can be used to detect accumulated anomalies. Specifically, we need to obtain the normal curve of the attribute value with time according to the model, and record the real-time data of the device to obtain the real-time curve. Then compare the similarity between the normal curve and the real-time curve in terms of average value and fluctuation. We improve the quartile interval algorithm based on statistics for the comparison process of curve similarity. The formula for calculating the range and slope similarity of two curves is as follows:

$$Sim_{value}(C_1, C_2) = \frac{1}{2} f(V_1, V_2) + \frac{1}{2} f(V_2, V_1)$$
$$Sim_{slope}(C_1, C_2) = \frac{1}{2} f(S_1, S_2) + \frac{1}{2} f(S_2, S_1)$$

$Sim_{value}(C_1, C_2)$ and $Sim_{slope}(C_1, C_2)$ are respectively the similarity of the two curve ranges and slopes, and the value range is [0,1]. The larger the value, the more similar the value and fluctuation of the two curves. V_1 and V_2 are 1*n vectors obtained by sampling the value of the curve, and S_1 and S_2 are 1*m vectors obtained by sampling the absolute value of the slope of the curve. $f(A, B)$ is a similarity evaluation method using some statistical values of two vectors, and the calculation formula is as follows:

$$\text{f(A,B)=} \begin{cases} 1, IQR_A = 0 and \bar{B} = \frac{Q_{1A}+Q_{3A}}{2} \\ 1 - \frac{\left|\bar{B} - \frac{Q_{1A}+Q_{3A}}{2}\right|}{IQR_A}, Q_{1A} \leqslant \bar{B} \leqslant Q_{3A} \\ 0.5 - \frac{Q_{1A}-\bar{B}}{3IQR_A}, Q_{1A} - 1.5IQR_A \leqslant \bar{B} < Q_{1A} \\ 0.5 - \frac{\bar{B}-Q_{3A}}{3IQR_A}, Q_{3A} < \bar{B} \leqslant Q_{3A} + 1.5IQR_A \\ 0, else \end{cases}$$

$\bar{B}$ is the average value of vector, Q_{1A} and Q_{3A} are the first quartile and third quartile of vector, that is, all the values in the vector are sorted and divided into four equal parts, and the 25% largest number and the 75% largest number are the first and third quartiles. IQR_A is the interquartile range of vector , and $IQR_A = Q_{3A} - Q_{1A}$. $f(A, B)$ compares the statistical value of B with the statistical value of A and then normalize it.

When the similarity is less than a certain threshold, it can be considered that the values or fluctuations of the two curves are significantly different, that is, abnormal data is detected. The larger the threshold value is selected, the more severe the condition that the data is judged to be normal.

C. Functional component anomaly

Functional component anomaly refers to the phenomenon that although the device can still work, the working state has not reached the normal level. As mentioned before, the hierarchical state automaton model can be used to detect abnormal conditions of device attributes. Some attributes of smart home devices are associated with functional components, so that when certain attribute values are detected to be abnormal, the device hardware that may be faulty can be found according to the correspondence between device attributes and functional components. Existing manuals, documents, and device data do not indicate the correspondence between hardware and attributes. So, when locating the cause of abnormal conditions caused by functional components fault, users need to participate in the process of determining the correspondence between functional parts and attributes. After users select the attributes required for modeling, they can map the name of the functional part with the related attributes, and the model records these information to obtain a complete correspondence.

D. Anomaly caused by human and environment

In addition to the abnormality caused by the failure of the device itself, there are other abnormal situations caused by human or environmental factors. The hierarchical state automaton model can be used to analyze the influence of time and attributes on abnormal results, and then to locate the possible abnormal causes.

First, we use the mode boundary and trend comparison methods to detect abnormal conditions, and at the same time, the position of the internal state of the model corresponding to the latest attribute data needs to be recorded for subsequent cause location analysis. Next, we can get the state link that causes the current anomaly. But there are some problems, for example, the current external state may be obtained by the conversion of multiple external states. During the backtracking process, it is necessary to determine which external state to trace back to. And there are multiple internal states in the external state and we need to determine which internal state to start the backtracking process. In order to solve the above problems, we propose an anomaly localization method based on probabilistic link backtracking. The external probability records the transition probability of each external state, which is reflected in the corresponding probability representation of each external edge, and is defined as follows:

$$P_{ij} = Count(o_{ij}) / \sum_{e_{kj} \in E} Count(o_{kj})$$

P_{ij} is the probability of the external edge e_{ij} that connects the external state S_i and S_j. $Count(o_{ij})$ is the number of operations o_{ij} performed. E is the set of external edges. That is, record the total number of operations converted to the external state, and the number of each operation, and calculate frequency to obtain the probability representation corresponding to the external edge.

The internal probability records the transition probability of each internal state to other external states, that is, the probability of the device changing to another mode when the device is running to the time point represented by the current internal state. It is reflected that each internal state has a corresponding probability representation, which is defined as follows:

$$P_k = JumpCount(I_k) / \sum_{e_{ij} \in E} Count(o_{ij}), I_k \in S_i$$

P_k is the transition probability of the internal state I_k in the external state S_i. $JumpCount(I_k)$ is the number of transitions of I_k. And $\sum_{e_{ij} \in E} Count(o_{ij})$ is the total number of device transitions from the external state S_i to all other external states.

Then we can perform a probabilistic backtracking process on the state according to the external and internal probabilities in the probability graph model. As shown in Figure 4, the backtracking is started from the state $S_2 t_3$. S_2 can be obtained from the two external state transitions S_1 and S_6. The backtracking is determined according to the size of the external probability, that is, 20% probability of backtracking to S_1, 80% probability of backtracking to S_6. Suppose that S_6 is used for backtracking, at this time the starting state is selected according to the internal probability, that is, 10% probability of backtracking from $S_6 t_1$, 20% probability of starting from $S_6 t_2$, and so on. In this way, we can determine the entire backtracking link.

After getting the state link of the current abnormal situation, we start to analyze the impact of the operation and time in the link on the abnormal attributes. As shown in Figure 4, in the state link, there is an edge between each two adjacent internal states. If the two internal states belong to two different external states, the operation information is recorded on the edge. Otherwise, time information is recorded on the edge. If the operation information has the greatest influence on the abnormal attribute, the possible cause of the failure is the corresponding operation error. If the piece of time information has the greatest influence on the abnormal attribute, the possible cause of the failure is an environmental mutation. If the accumulation of multiple pieces of time information has the greatest impact on abnormal attributes, the possible cause of the failure is that the cumulative running time of the device is too long.

V. System Implementation

We design and implement a smart home device modeling tool and anomaly detection system. The overall architecture of the tool and system is shown in Figure 5.

Modeling tool is responsible for the construction and update process of the model, including basic components such as

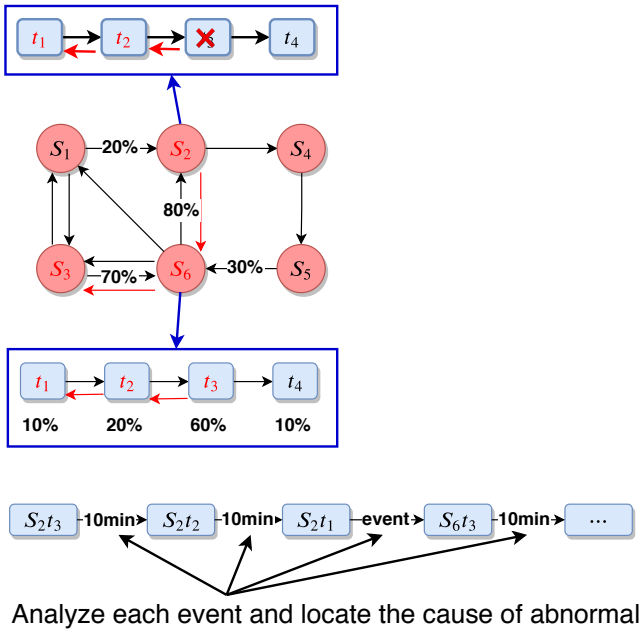

Fig. 4. Probabilistic link backtracking.

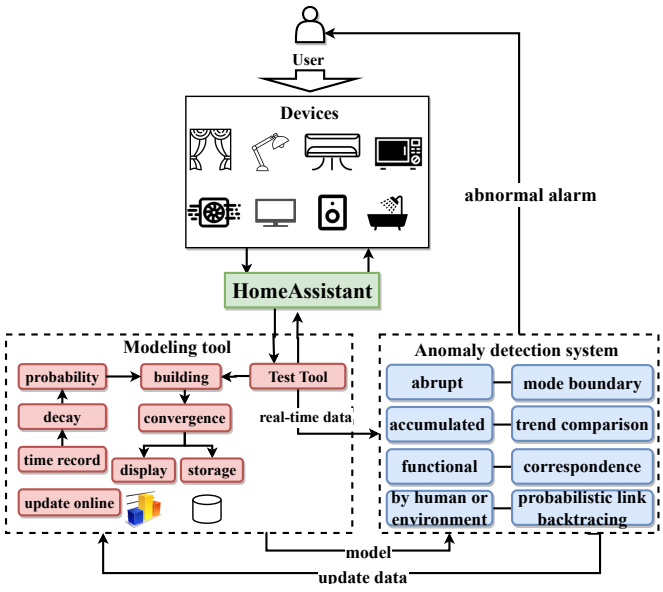

Fig. 5. Architecture of the tool and system.

model representation, convergence judgment of the model, model visualization and storage, and model optimization mechanisms such as probability graph model construction, decaying mechanism and different time recording methods . In addition, it also includes a test tool for regularly obtaining device attributes and calling operations through the smart home control platform. Specifically, it includes some scripts that interact with the smart home control platform, operation and attribute filters, timers, data processing and log information generation processes.

The anomaly detection system includes the device anomaly

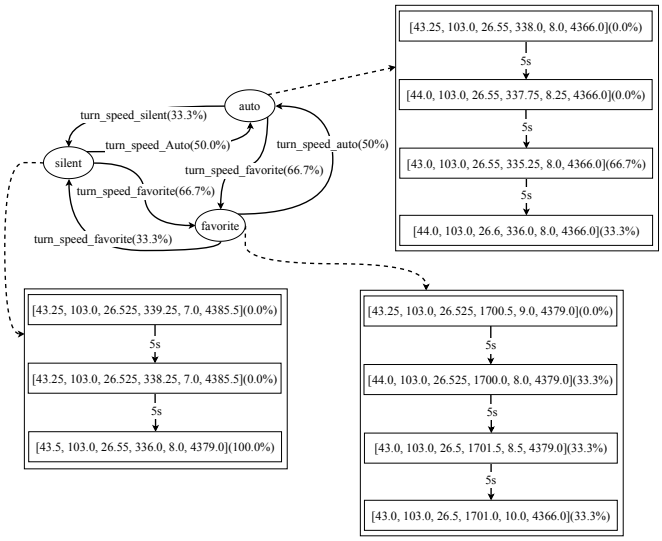

Fig. 6. Model of the purifier.

detection methods mentioned before, which can subdivide anomaly detection situations. Each situation corresponds to a different method and implementation mechanism.

VI. Evaluation

A. Experimental devices

There are many types of smart home devices, which involve all aspects of life. We select two widely-used devices, including Mi Air Purifier 2S and Mi Sterilization Humidifier as experimental devices, both of which include multiple modes and attributes. In addition, the water mist generated by the humidifier can act as an environmental change on the air purifier, which can meet a variety of experimental needs.

B. Modeling experiments

The purpose of the modeling experiment is to build hierarchical state automaton model according to the behavior path of devices. In the experiment, the system parameters are set as follows: the time interval for obtaining attributes is 5 seconds, the time interval for calling operations is 90 seconds, and the decay factor is 0.5.

During the modeling process, the attributes and operation sets of devices will be obtained and filtered. For the air purifier, the original attribute set includes 28 attributes, and the attribute filtering method mentioned before is used to reduce the number of attributes to 14. Users can also manually select some attributes from the filtered attribute set. At the same time, three operations are selected for subsequent modeling process.

The model display of air purifier is shown in Figure 6, where we select 6 of 14 attributes. Each mode corresponds to an external state, the edge between the external states corresponds to the operation, and the internal state is represented by the attribute state vector and updated in a decaying way. The external probabilities are displayed on the external edges, and the internal probabilities correspond to each internal state, and both will constantly change and update. The schematic

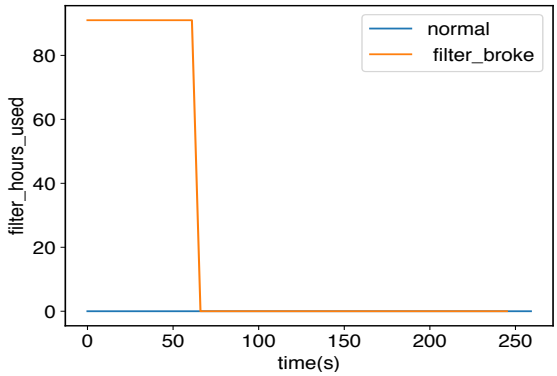

Fig. 7. Normal and abnormal filter_hours_used.

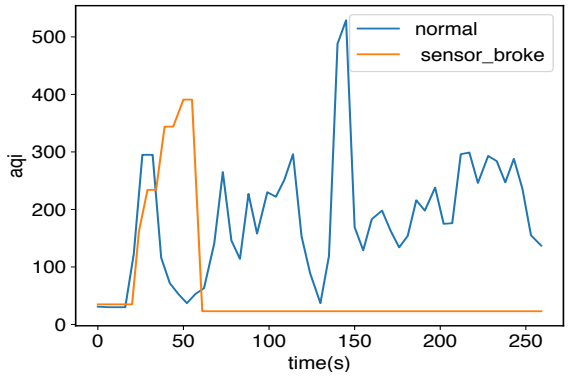

Fig. 8. Normal and abnormal aqi.

diagram of the model is generated under specific settings and hardware versions, but the model structure it represents is suitable for any smart home device.

C. Anomaly detection experiments

In the anomaly detection experiments, we use anomaly injection to verify the accuracy of anomaly detection and location functions. Specifically, during the runtime of the air purifier in auto mode, the humidifier is turned on to generate water mist to let the air purifier run in a polluted environment, and then artificially add some abnormalities to the air purifier. In this experiment, the threshold of the value and the slope in cumulative anomaly detection is set to 0.25, the number of backtracing steps is set to 10, and the threshold of the judgment of the abnormal mutation is 0.2.

Figure 7 shows the *filter_hours_used* attribute data obtained in the normal state recorded in the model and the abnormal state after removing filter. First, we use the model boundary to detect the attribute. After analysis, the value range of *filter_hours_used* is a linear change. The real-time data is not within mode boundary, and it can be determined as a sudden change. Since the *filter_hours_used* attribute has a corresponding relationship with the functional part filter, it is possible to locate the filter anomaly and achieve the goal of anomaly detection and localization.

Figure 8 shows the *aqi* attribute data obtained in the normal state recorded in the model and the abnormal state after the air quality sensor is removed. The data value of the normal state analyzed is the fluctuation type, so only the upper and lower boundaries of the data need to be recorded. However, the value of real-time data is within the mode boundary obtained in the normal state. The value similarity of the two curves is 0.18, and slope similarity is 0.24, and both are less than the corresponding threshold. Since the *aqi* attribute has a corresponding relationship with the air quality sensor, the abnormal cause can be located to the abnormality of the air quality sensor to achieve the targets of anomaly detection and localization.

Figure 9 shows the *motor_speed* attribute data obtained in the normal state recorded in the model and the abnormal state after blocking in the air inlet. The analysis shows that the data is in a fluctuating state, and only the maximum and minimum values of the two need to be recorded. The value range of real-time data is always between the upper and lower boundaries of the model data, so it does not exceed the model boundary. The value similarity of the two curves is 0.21, which is relatively low and less than the corresponding threshold. Since the *motor_speed* attribute has no corresponding relationship with the functional components, it is necessary to use the probabilistic link backtracking algorithm to continue the analysis.

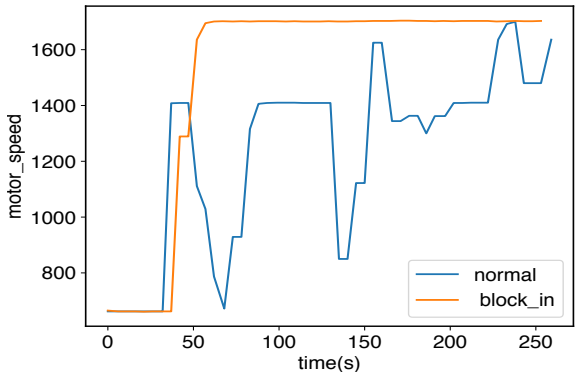

Fig. 9. Normal and abnormal motor_speed.

In addition to detecting abnormal conditions, anomaly detection methods should also avoid the occurrence of false alarms, that is, for a normally running device, its data should not be detected for abnormal conditions. Figure 10 shows the *motor_speed* attribute data of two normal states. In terms of mode boundaries, the value of real-time data after analysis is within the mode boundaries of model data. The similarity between the two values and the absolute value of the slope is 0.63 and 0.71. There are no detected abnormalities, and the result of the experiment meet the expected goal.

VII. Related Work

Device modeling methods. In the industrial field, digital twin is the commonly used modeling method, which was first

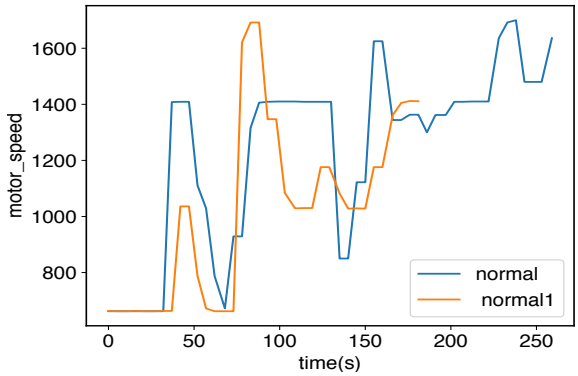

Fig. 10. Normal motor_speed.

proposed by Michael Grieves of the University of Michigan in 2002 [16]. Products, factories or business services can all have corresponding digital twin models [6]. According to Gartner's data, by 2021, nearly half of large industrial companies will use digital twin technology to improve system performance, technical performance, and technical risk assessment, while improving system effectiveness and system risk by about 10% [8]. In the construction field, Building Information Modeling (BIM) is a modeling technology used to reduce project costs, increase productivity and quality, and shorten project delivery time [7]. The building information model simulates a building project in a virtual environment and can digitally build an accurate virtual model of a building. In the computer field, modeling methods can be divided into system modeling, data modeling, and program modeling. The system model provides knowledge about the system in a certain form (such as text, symbols, charts, objects, mathematical formulas, etc.) [12]. Data modeling is an abstract organization of various types of data in the real world. According to the different uses of the model, it can be divided into database model and algorithm model. Program modeling is mainly used in the program analysis process within the system. The process of program analysis can be divided into static analysis and dynamic analysis [19].

Anomaly detection methods. The focus of existing work [13][14] is mainly about discovering abnormal user behaviour through smart home device data, which may affect the device or the user itself. The problem with the work is that they only focus on abnormal conditions of user behaviour, and ignores abnormal conditions caused by environment, time, and device itself. And it does not deeply analyze the hardware, software, and data information of devices and cannot accurately detect specific abnormal conditions. Common anomaly detection algorithms include statistical methods, clustering, integrated learning and deep learning, such as the 3-Sigma principle and the interquartile range algorithm in statistical models, K-means and DBSCAN algorithm [10] in clustering, Isolation Forest algorithm [15] in the integrated learning model and the deep neural network and autoencoder in the deep learning model. In terms of anomaly localization, common methods include the fault tree method based on graph theory [18], the expert system [17], and machine learning-based neural networks, SVM [9] and other algorithms.

VIII. Conclusions

Today's modeling methods are not applicable to the diversity of smart home devices and the complexity of anomaly, which can't accurately express the behaviour of smart home devices. In this paper, we propose a smart home device modeling method and anomaly detection methods based on hierarchical state automata, and at the same time build a modeling tool and anomaly detection system. Our experiments show that the methods can build the behaviour model of the devices, and locate various anomalies accurately.

IX. Acknowledgements

This work is in part by Huawei HIRP under Grant HO2018125086, and Youth Innovation Promotion Association at CAS. Lijie Xu and Wei Wang are the corresponding authors {xulijie, wangwei}@otcaix.iscas.ac.cn.

References

[1] Forrester Analytics: Smart Home Devices Forecast, 2018 To 2023 (US). https://www.forrester.comhttps://www.forrester.com.
[2] Google Assistant. https://assistant.google.comhttps://assistant.google.com.
[3] Home Assistant. https://www.home-assistant.iohttps://www.home-assistant.io.
[4] HomeKit. https://developer.apple.com/homekithttps://developer.apple.com/homekit.
[5] Mi BBS. http://bbs.xiaomi.cn/t-1396972http://bbs.xiaomi.cn/t-1396972.
[6] What is digital twin technology-and why is it so important?
[7] M. J. Y. N. Azhar S, Nadeem A. Building information modeling (bim): A new paradigm for visual interactive modeling and simulation for construction projects. In *First International Conference on Construction in Developing Countries*, 2008.
[8] P. C. Prepare for the impact of digital twins. *Gartner: Stamford, CT, USA.*, 2017.
[9] V. V. Cortes C. Support-vector networks. *Machine learning.*, 20(3):273–297, 1995.
[10] S. J. Ester M, Kriegel H P. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.
[11] D. Evans. The internet of things: How the next evolution of the internet is changing everything. cisco white paper. 2011.
[12] W. X. L. Guo Q S, Yang X Y. System modeling. 2006.
[13] H. K. Z. Guralnik V. Learning models of human behaviour with sequential patterns. In *Proceedings of the AAAI-02 workshop "Automation as Caregiver*, pages 24–30, 2002.
[14] S. D. Kang W, Shin D. Detecting and predicting of abnormal behavior using hierarchical markov model in smart home network. In *2010 IEEE 17Th International Conference on Industrial Engineering and Engineering Management. IEEE*, pages 410–414, 2010.
[15] Z. Z. H. Liu F T, Ting K M. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining. IEEE*, pages 413–422, 1996.
[16] G. M. Digital twin: manufacturing excellence through virtual factory replication. *White paper.*, 29:1–7, 2014.
[17] J. P. Introduction to expert systems. In *Addison-Wesley Longman Publishing Co.* 1998.
[18] L. P. An application of fault tree analysis to the identification and management of risks in government funded human service delivery. In *Proceedings of the 2nd International Conference on Public Policy and Social Sciences held in Kuching*, 2011.
[19] X. J. X. Y. W. Q. L. B. L. L. D. W. C. Z. C. L. C. Y. Zhang J, Zhang C. Recent progress in program analysis. ruan jian xue bao/journal of software. *Journal of Software.*, 30(1):80–109, 2019.