Self-Adaptive software changes analysis method based on “Detection-Recognition” Mechanism

He Zhang, Qingshan Li*, Lu Wang*, Wen Cheng
Department of Software Engineering
Xidian University
Xi’an, P. R. China
qshli@mail.xidian.edu.cn

Abstract—**Self-Adaptive Systems (SASs) need to analyze software changes accurately and continuously, that is, recognize events caused by changes, and adjust structure or behavior. However, present event recognition methods frequently monitor events, resulting in waste of system resources. And most of them ignore the impact of operating environment uncertainty, causing errors in recognizing the event and directly affecting the reliability of SASs. Addressing the above problems, this paper proposes an event recognition method based on "detection-recognition" mechanism. Firstly, the Naive Bayesian Classification algorithm is used to detect the state of the system. If the system is judged to be abnormal, we will combine with rule reasoning and fuzzy reasoning to recognize events. The system does not have to monitor the occurrence of events from time to time, avoiding the waste of system resources. Moreover, the probabilistic reasoning method of Bayesian Classification and the introduction of fuzzy reasoning can cope with environmental uncertainty and improve the accuracy of event recognition. Finally, we exemplify this mechanism with the Web system, which proves the effectiveness of the methods.**

Keywords-component; Self-Adaptive software; Analyze; Naive Bayes Classification; Rule reasoning

I. INTRODUCTION

Self-adaptive Systems (SASs) modify their behaviors or structures in response to their perception of the environment and the system itself [1]. Adaptation process for SASs generally includes monitor, analyze, plan and execution [2]. The **Analyze** is responsible for judging whether the system needs to be adjusted by observing the changes information of the Monitor, and to recognize accurately the events triggered by the changes.

At present, there are few researches on changes analysis. Some researchers apply **ontology** to analyze system situation [3][4]. The accuracy of ontology reasoning is higher, but the ontology has high demand for developers, and it is difficult to perform dynamic correction during system running, so we will not consider the ontology reasoning method. Some researchers use **rule** methods to recognize environmental events [5], and some recognize active database events [6]. The types of events recognized in the above studies are **single or limited**. Nowadays, the operating environment or system structure is highly likely to change. If the ontology or rules are **unchanged**, the **accuracy** of event recognition will be low.

There are other methods for recognizing events, including event listeners [7], a generalized modeling framework of fault detection and correction processes [8]. Most methods lack research on system state detection, they usually recognize events directly. However, software changes do not necessarily trigger events, for example, fluctuations and surges of system status data will not affect the normal provisioning of system functions and system status, so they will not evolve into events as the system runs. Such frequent monitoring of events will frequently make use of system resources, resulting in **waste of resources**.

In addition, if the event is recognized incorrectly, even if the system performs a series of adjustments, it may not achieve the expected results, or even make the system crash, which seriously affects the **reliability** of the SASs. Nowadays, the dynamic operating environment of complex software and the complexity of its structure cause the process of event recognition faces **uncertainties** such as environmental complexity and ambiguity of demand. Most methods focus on recognizing events in a certain environment, its accuracy cannot be guaranteed in the dynamic and variable environments. The present ideas of processing uncertainty mainly include **fuzzy logic** and **probability theory** [9][10]. These methods only consider the research of design phase, or have specific scenario constraints.

In response to above issues, this paper proposes a “**detection-recognition**” mechanism, which first judge the system state, if it is abnormal, then recognize the event. In the “detection” stage, we establish the **Naive Bayesian Classification** model to judge system status quickly by analyzing the probability value. In the “recognition” stage, we combine the **rule reasoning** and **fuzzy reasoning** to recognize the event, which can improve the accuracy of event recognition and migrate this method to other systems through the addition and modification of rules.

This paper is organized as follows: section Ⅱ provides the detail of our event recognition method; section Ⅲ introduces our experiment and some discussions; conclusion is discussed in section IV.

DOI reference number: 10.18293/SEKE2019-049

II. The Event Recognition Method

We propose the "detection-recognition" mechanism to analyze events triggered by the changes. It contains two stages of abnormal state detection and event recognition.

A. *Abnormal State Detection*

Abnormal state definition: the system's functional or non-functional requirements are affected due to system events. To detect accurately and quickly the abnormal state of the system running, we apply the **Naive Bayesian Classification** model to judge the system status.

First, we convert the numerical data collected in the system log into character data by conversion threshold. The processed log is divided into the training set and the test set by *the 4-fold cross-validation method*.

Second, the frequency of occurrence of features or categories in the training set is counted to estimate the probability of occurrence. We adopt the *Laplace transformation method* to avoid the situation where the probability value is 0.

Then, we get the error rate of the model by operating the test set. If the error rate is higher than the preset tolerance, we will return to data processing process, and dynamically adjust the threshold of the converted numerical data. In general, the tolerance is set to 8%, this value will be verified in Section Ⅲ.

Finally, the model calculates the system state, as in (1).

$$P(Category \mid FeatureValue) = \frac{(\prod_{i=1}^{n} P(FeatureValue_i \mid Category)) \times P(Category)}{\prod_{i=1}^{n} P(FeatureValue_i)} \tag{1}$$

As in (1), *FeatureValue* refers to the eigenvalues that can characterize the state of the system. *Category* is divided into *Normal* and *Abnormal* categories in this paper. When P(*Abnormal*|*FeatureValue*) is greater than P(*Normal*|*FeatureValue*), we consider that the current system is in an abnormal state, and further need to inference the event.

B. *Event Recognition*

We combine the rule-based reasoning and fuzzy reasoning to recognize events. The working process is shown in Fig.1. We first establish the events library. Then, according to the predefined recognition rule base, the event information that occurs is reasoned. At the same time, we apply fuzzy reasoning to supplement rule reasoning, further achieve feedback and correction of the rule base and ensure the accuracy of event recognition.

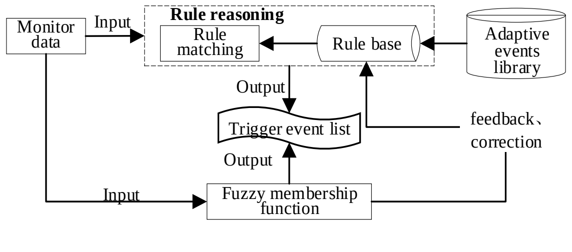

Fig.1. Event recognition method working process

1）Event recognition method based on rule reasoning

The process of this method is as follows. First, we define the corresponding mapping rules for the event. The rules indicate the relationship between system status and events, as in (2).

rule *RuleName*
when *Judging condition，* ***then*** *Event information (&& action)*
end (2)

We introduce the mapping rules between system states data and " Unit Fault" as an example, as shown in Fig.2.

```
rule UnitFault
    when SystemState (heartBeatInterval>threshold&&
           responseTime>threshold&&errorRate>threshold
           &&nodeState== "normal" )
    then <E0201> && getTime();
end
```

Fig.2. UnitFault rule example

Among them, the rule conditions are the judgment of the system status value, and the latter part of the rule is the event ID and the time that successfully matches the rule.

Then, we recognize events based on the mapping rules. We match the status information with the conditions of the rules in the rule base. If the current status information can match multiple rules, the rules will be placed in the conflict set. Conflict resolution strategies such as predefined rule priorities or definition rule groups are used to resolve conflicts between rules in conflict sets. Once the conflicts are complete, the rules will be executed in order. Then we will output event information or perform corresponding actions.

This method belongs to the category of precise matching, so events that have occurred can be accurately inferred according to the rules.

2)Event recognition method based on fuzzy reasoning

In this method, we establish fuzzy sets and membership function for the state eigenvalues of the system. Then, we establish fuzzy rules. In the rule base, the rules include the form of the fuzzy set in addition to the above-mentioned form of passing the threshold. Finally, the matching degree between the current system state and each rule is calculated by (3). We select a rule with the largest matching degree, output event or perform action in the latter part of the rule.

$$MatchingDegree_R = \sum_{i=1}^{n} (membership_i \times weight_i) \tag{3}$$

$MatchingDegree_R$ indicates the matching degree between the system state and rule *A*. *n* indicates the number of system state eigenvalues. $membership_i$ indicates that the eigenvalues *i* belongs to the membership of the fuzzy set of eigenvalues in rule *R*, and $weight_i$ indicates the weight corresponding to the *i-th* eigenvalues.

In summary, the event recognition method based on fuzzy reasoning mainly supplements and corrects rule reasoning to improve the accuracy of event recognition.

III. Experiment

To validate the methods of this paper, we choose *BookStore* System as the case to test the ability and accuracy of "detection-recognition" mechanism.

A. *Bookstore system*

BookStore is a e-commerce system that uses the B/S architecture to provide users with functions such as registration login, product browsing, product payment and so on. Various types of events such as server corruption, response timeout, network bandwidth change, etc. may occur during system running. And the user requirements, computing resources, system overhead, etc. in the system are easily affected by the open environment, it is not possible to define recognition rules for all events during the design phase. Therefore, *BookStore* can be used to test the ability of this method to recognize multiple event types and uncertain event.

B. *The experiment for recognizing events*

*1)**Model system status***. When detecting the system status, we use the three characteristic values of ***response time***, ***page error rate*** and ***load*** to characterize the system status. The node load is calculated by (4).

$$nodeLoad = 0.4 * CPU + 0.3 * memory + 0.3 * disk \tag{4}$$

*2)**Establish event library***. We use tuples to represent event and store it in the event library to facilitate event information output during subsequent event recognition, as in (5).

$$Event = \{ E\_Id, E\_Name, E\_Value, E\_Time, E\_Effect, E\_Priority, E\_Duration \} \tag{5}$$

E_Id is composed of 4 digits. The first two digits indicate the event type. "01" refers to the type of node resource changes."02" refers to the type of unit resource changes such. "03" refers to the type of changes in the business logic layer. "04" refers to the type of communication environment changes. "05" refers to the type of hardware environment changes. The last two digits indicate specific events under a particular type.

E_Effect can be divided into local and global categories. The local effect refers to an event that affects only one software unit, and the global effect refers to an event that affects the global system. *E_Duration* indicates the duration of the event from being recognized to the current time, and it can be used as a reference to set the sequence of event processing.

*3)**Experiment Design and result***

We continuously collect and store the running data of the Bookstore, and we set the dataset size as follows: the size of data set 1 to set 6 is 300, 500, 700, 900, 1100, 1300 data, respectively. The following tests are performed on a computer with Inter(R) Core(TM) i5-4570 processors and 8GB RAM.

We use the "detection-recognition" mechanism to detect the state of the *BookStore* system over a period of time and to recognize events, as shown in Table Ⅰ.

TABLE Ⅰ

The System Status And Event Display Of Bookstore

System status	Id	Name	Effect	Time	Priority
abnormal	0101	User server overload	global	11:28 2018-10-15	urgent
abnormal	0501	Home response timeout	global	12:05 2018-10-15	urgent
abnormal	0502	Reduce ads	local	12:43 2018-10-15	general
abnormal	0201	Product display page lost	local	13:18 2018-10-15	very urgent
abnormal	0301	Network delay	global	13:47 2018-10-15	urgent

We show the main attributes of the event. The priority refers to the urgency of the event to be processed, which is determined by the impact of the event on the functional and non-functional requirements of the system. We set the priority of events that have a large impact on functional or non-functional requirements to be *very urgent*, such as server damage events. The priority of a more influential event is set to *urgent*, such as response timeout. The priority of the less influential event is set to *general*, such as the user request to reduce the number of ads.

From Table Ⅰ we can see that the mechanism of this paper can detect the state of the system and further recognize various types of events. It shows that the mechanism can effectively realize the main tasks of the analyze of the adaptive process.

To verify the time efficiency of the mechanism, we test the time when it processes data sets of different sizes, as shown in Fig.3. As the data size increases, the operation time of the mechanism increases, but the overall does not exceed 2500ms, indicating that the mechanism has higher time efficiency.

To verify the accuracy of the mechanism, we test the error rate of abnormal state detection under the aforementioned data sets, as shown in Fig.4. Meanwhile, the Accuracy, Precision and Recall of the method are verified under the aforementioned data sets, as shown in Table Ⅱ.

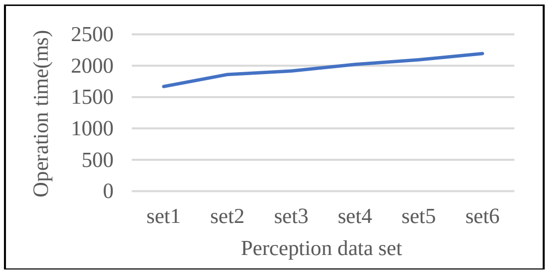

Fig.3. Operation time of the mechanism

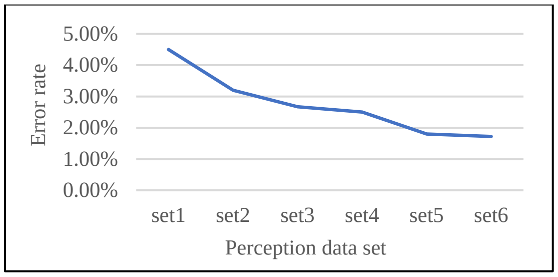

Fig.4. Error rate of state detection method

TABLE Ⅱ
ACCURATENESS OF EVENT RECOGNITION METHOD UNDER DIFFERENT DATA SIZES

Set number	Accuracy	Precision	Recall
Set 1	95.33%	99.61%	95.27%
Set 2	96.2%	99.54%	96.24%
Set 3	96.85%	99.67%	96.88%
Set 4	97.33%	99.63%	97.45%
Set 5	97.64%	99.70%	97.75%
Set 6	97.76%	99.67%	97.94%

As shown in Fig.4, as the data size increase, the error rate of the detection method is declining and gradually gradual. Since the effect of the Bayesian model depends not only on the amount of training data, but also on the construction of the classifier and the characteristics of the data to be classified, there are inevitable errors in the method for state detection. As shown in Table Ⅱ, the Accuracy, Precision and Recall of the event recognition method are higher in the data sets of different scales, indicating that the method proposed in this paper has a better recognition effect.

In the state detection, to select the appropriate error rate tolerance, we set different tolerances when realizing detection method. Then we recognize events under dataset 6 to obtain the accuracy of recognition, as shown in Fig.5. The accuracy has a significant continuous decline when the tolerance is greater than 8%. At 2% to 8%, the decline is lower. If the tolerance is smaller, the system will constantly adjust and cause system overhead. Therefore, we take 8% as the tolerance.

We compare the Recall of using only rule reasoning and our method under the aforementioned data set, as shown in Fig.6. The Recall of this method is higher than that of rule-based reasoning. Fuzzy reasoning supplements the rule reasoning when encountering an unknown situation. This also verifies the effectiveness of the event recognition method in this paper.

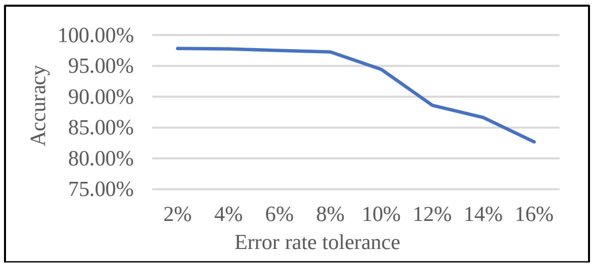

Fig.5. Impact of error rate tolerance on event recognition accuracy

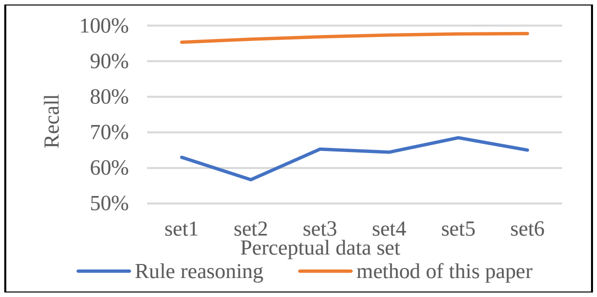

Fig.6. Comparison of event recognition methods

IV. CONCLUSION AND FUTURE WORK

In this paper, we propose a "detection-recognition mechanism, which can effectively avoid the waste of system resources, and cope with the uncertainty in the environment, so that the accuracy of event recognition is improved. In the future, we will further observe the operating characteristics of the system, and consider the online dynamic correction method of the rules. And we will expand the type of recognition event to further enhance the range of recognized events.

ACKNOWLEDGMENT

This work is supported by the Projects (61672401) supported by the National Natural Science Foundation of China; Projects (315***10101, 315**0102) supported by the Pre-Research Project of the “Thirteenth Five-Year-Plan” of China.

REFERENCES

[1] Lemos R D, Giese H, Müller H A, et al. Software Engineering for Self-Adaptive Systems: A Second Research Roadmap[J]. Lecture Notes in Computer Science, 2013, 5525:1-32.

[2] Frank D. Macías-Escrivá,Rodolfo Haber,Raul del Toro,Vicente Hernandez. Self-adaptive systems: A survey of current approaches, research challenges and applications[J]. Expert Systems With Applications,2013,40(18).

[3] Baader F. Ontology-Based Monitoring of Dynamic Systems[J]. 2014.

[4] Paola A D. An Ontology-Based Autonomic System for Ambient Intelligence Scenarios[M]// Advances onto the Internet of Things. Springer International Publishing, 2014:1-17.

[5] C. K. Chang, K. Oyama, H. Jaygarl and H. Ming, "On Distributed Run-Time Software Evolution Driven by Stakeholders of Smart Home Development (Invited Paper)," 2008 Second International Symposium on Universal Communication, Osaka, 2008, pp. 59-66.

[6] Jin Y. Management of composite event for active database rule scheduling[C]// IEEE International Conference on Information Reuse & Integration. IEEE, 2009:300-304.

[7] J. Lang, M. Jantošovič and I. Polášek, "Re-usability in complex event pattern monitoring," 2012 IEEE 10th International Symposium on Applied Machine Intelligence and Informatics (SAMI), Herl'any, 2012, pp. 265-270.

[8] Okamura H, Dohi T. A Generalized Bivariate Modeling Framework of Fault Detection and Correction Processes[C]// IEEE, International Symposium on Software Reliability Engineering. IEEE Computer Society, 2017:35-45.

[9] Yang Q , Jian Lü, Li J , et al. Toward a fuzzy control-based approach to design of self-adaptive software[M]. 2010.

[10] Xu L , Wang X L , Wang X F . Fast Method of Compound Event Probability Calculation Based on Binary Tree[C]// Fifth International Conference on Natural Computation. IEEE Computer Society, 2009.