Behavior Trend Analysis Method of Software Runtime Environment Elements Based on ARIMA

Qiuying Li¹², Minyan Lu¹² and Tingyang Gu¹²,*

¹School of Reliability & Systems Engineering, Beihang University, China
²Science & Technology on Reliability & Environmental Engineering Laboratory, Beijing, China

*Corresponding author email: li_qiuying@buaa.edu.cn

Abstract. Behavior trend analysis method based on auto-regressive integrated moving average (ARIMA) is presented for software runtime environment elements (REEs). The history data of environment element behavior is collected to build behavior trend analysis model. Then, behavior trend of environment elements is predicted before the environment elements actually change. Thus, according to the analysis results, we can determine the corresponding adaptive strategy before the environment elements actually change, which provides a reference for the selection and implementation of active adaptive strategy for the self-adaptive software.

Keywords: Runtime environment; Environment elements; Behavior trend; ARIMA; Trend analysis.

1. Introduction

Self-adaptive software systems are one kind of systems which can adapt themselves with changing runtime environment by adjusting their behavior and structures. Existing self-adaptive software systems mostly use ECA (event: condition→operation) to respond to these changes of the REEs, i.e. when it is detected that the REEs have changed and reached their thresholds, the corresponding strategies are triggered according to the pre-set rules, and the adaptation will be performed under the drive of these strategies.

However, the existing methods have the following shortcomings: firstly, the adaption starts only after the REEs have been detected to change. Secondly, due to the uncertainty of changes in the runtime environment of self-adaptive software, designers cannot accurately describe all possible future changes and how to change in the design phase. Therefore, it is difficult to describe them clearly in the design phase. As a result, the rules set in advance may not be able to meet the needs of changing REEs in operating phase.

The existing researches on self-adaptive software environment mainly focus on the following aspects. First, how to integrate the REEs into the development model of adaptive software [1-3]. Second, how to associate the environment with the requirements of adaptive software, and better analyze and judge the evolution of requirements [4-5]. Third, how adaptive software responds when changes in the environment are detected [6], i.e. how the ECA method [7] associates environment changes with adaptation strategies. In terms of environment modeling, only several individual environment elements are modeled, and the descriptions of REEs are not systematic. Moreover, they pay attention to the impact of adaptive software responding to the environment which has changed, but lack of prediction of how the environment will change [8].
Time series analysis method is one kind of methods that analyze historical time series data and find rules from them to predict the future data changes. There are many forecasting algorithms based on time series. Currently, ARIMA model and BP neural network model are two widely used models [9]. This paper proposes a behavior trend analysis method based on ARIMA for the software REEs. Based on the historical data of REEs’ behavior, their future behavior trends are analyzed and predicted. Before the actual changes of REEs take place, adaptation strategies can be formulated in advance.

2. Behavior Trend Analysis Method of Software REEs Based on ARIMA

2.1. Software Runtime Environment Elements

Here software REEs are divided into computing environment elements and user environment elements. Computing environment elements include bandwidth, memory, CPU usage, load etc., which are related to computing, and user environment elements include the number of visits, refresh frequency, etc., which are related to the users’ usage characteristics.

Table 1. Computing environment elements.

| Name                               | Environmental entity | Category             | Acquisition method         | Behavior                          | Predictability |
|------------------------------------|----------------------|----------------------|-----------------------------|-----------------------------------|----------------|
| Amount of free memory             | RAM                  | Hardware resources   | Hardware register           | Value change event (rise, fall)   | yes            |
| Memory usage                       | RAM                  | Hardware register    |                             | Value change event (rise, fall)   | yes            |
| Maximum memory address             | CPU                  | Hardware resources   | Hardware register           | Value change event (rise, fall)   | yes            |
| Minimum memory address             | CPU                  | Hardware resources   | Hardware register           | Value change event (rise, fall)   | yes            |
| Number of CPU instructions         | server               | System configuration files and monitors |                              | Value change event (rise, fall)   | yes            |
| Number of idle servers             | server               | System configuration files and monitors |                              | Value change event (rise, fall)   | yes            |
| Server load average                | server               | System configuration files and monitors |                              | Value change event (rise, fall)   | yes            |
| Number of free hard drives         | hard disk            | Hardware monitor     | Value change event (rise, fall) | no                                | no             |
| Hard disk I/O speed                | hard disk            | Hardware monitor     | Value change event (rise, fall) | no                                | no             |
| Bandwidth usage                    | network              | Internet resources   | Network monitoring equipment | Value change event (rise, fall)   | yes            |
| Data throughput                    | network              | Internet resources   | Network monitoring equipment | Value change event (rise, fall)   | yes            |
| Number of TCP connections          | network              | Internet resources   | Network monitoring equipment | Value change event (rise, fall)   | yes            |
| Network bandwidth                  | network              | Internet resources   | Network monitoring equipment | Value change event (rise, fall)   | yes            |
| Network transfer speed             | network              | Internet resources   | Network monitoring equipment | Value change event (rise, fall)   | yes            |
| Network delay                      | network              | Internet resources   | Network monitoring equipment | Value change event (rise, fall)   | yes            |
| Number of processes                | processes            | Operating system resources | Operating system interface | Value change event (rise, fall)   | no             |
| Number of threads                  | threads              | Operating system resources | Operating system interface | Value change event (rise, fall)   | no             |
| Number of events                   | events               | Operating system resources | Operating system interface | Value change event (rise, fall)   | no             |
| Number of available services       | service              | Service resources    | Discovery server            | Value change event (rise, fall)   | yes            |
| Service call time                  | service              | Service resources    | Monitoring software         | Value change event (rise, fall)   | yes            |
| Service call frequency             | service              | Service resources    | Monitoring software         | Value change event (rise, fall)   | yes            |
| Service response time              | service              | Service resources    | Monitoring software         | Value change event (rise, fall)   | yes            |

The changes of the REEs include value-based changes and status-based changes. Value-based changes can be CPU usage, memory usage, etc., and status-based changes can be disk drive, network card drive, etc. As the user environment elements change, the computing environment elements will change accordingly, so we only analyze the behavior trends of the computing environment elements.

Table 1 lists the computing environment elements.

2.2. Method Steps

The flow chart of this method is shown in figure 1.
Select environment elements according to the environment element system

Collect historical data of the environment elements

Differential processing of historical data series

Perform autocorrelation analysis and partial autocorrelation analysis

Parameter estimation and model fitting

Significance test

Residual analysis

Use models to predict and analyze changes in environment elements

**Figure 1.** The flow chart of behavior trend analysis method based on ARIMA.

**Step 1:** Select software REEs.
**Step 2:** Collect historical data of the behavior of REEs and draw historical data sequence charts. Tools can be used that comes with the system, or auxiliary tool plug-ins installed in the system or programs coded for collecting data. The plug-ins should have little impact on the system and not affect the data accuracy.
**Step 3:** Perform differential processing on historical data and remove the periodicity of data trend. Generally the mean values of original time series fluctuate up and down, which do not meet the need of stationarity and need to be processed with difference processing. So they can be processed by the first-order or higher-order difference successively, then the time series are judged whether stationary after difference.
**Step 4:** Perform autocorrelation analysis and partial autocorrelation analysis based on the historical data sequence after difference processing. The calculation of autocorrelation function ACF is given as follows.

\[
\rho_k = \frac{\text{Cov}(y_t, y_{t-k})}{\text{Var}(y_t)} = \frac{\gamma_k}{\gamma_0}
\]  

(1)

where \(\text{Cov}(y_t, y_{t-k})\) and \(\gamma_k\) represent autocovariance, \(\text{Var}(y_t)\) and \(\gamma_0\) represent sample difference.

The calculation of autocorrelation function PACF is given as follows:

\[
\rho_{y_{t-k}|y_{t-k+1}} = \frac{E\left[\left(y_t - \hat{E}y_t\right)\left(y_{t-k} - \hat{E}y_{t-k}\right)\right]}{E\left(y_{t-k} - \hat{E}y_{t-k}\right)^2} = \frac{E[y_t - \hat{E}y_t]}{E[y_{t-k} - \hat{E}y_{t-k}]}
\]  

(2)

where \(\rho_{y_{t-k}|y_{t-k+1}}\) represents the influence of \(y_{t-k}\) on \(y_t\) after eliminating the interference of \(k-1\) random variables \(y_{t-1}, y_{t-2}, \ldots, y_{t-k+1}\) in the environment element data, and \(\hat{E}y_t = E[y_t|y_{t-1}, \ldots, y_{t-k+1}], \hat{E}y_{t-k} = E[y_{t-k}|y_{t-1}, \ldots, y_{t-k+1}]\).

**Step 5:** Estimate the model parameters and construct a trend analysis model of REEs behavior based on the ACF and PACF functions. The model parameters can be estimated by means of the least square method or the maximum likelihood method, and a set of optimal \(p, q\) are finally obtained as the \(p, q\) combination in the ARIMA\((p,d,q)\) model.

**Step 6:** Implement residual analysis and significance test. The residual analysis of the model is to verify whether the residuals of the model are independent, that is, whether the residual sequence is a white noise process with mean value of 0 and variance of 1.
Step 7: Analyze the behavior trend of environmental elements. Based on the constructed model, for any given time, the values of REEs can be calculated, then the behavior trend of REEs can be predicted.

3. Experiment and Analysis

3.1. Experiment Environment Deployment
Rainbow-znn \cite{10} software is selected as the target system for instance verification. The deployment of the entire experimental environment is shown in figure 2.

![Figure 2. The deployment diagram of experiment environment.](image)

1) Web1 to web4 services and Apache-2.2.16 are deployed on node 4, 5, 6 and 7 respectively, so they can simulate the service pools with service 1 to service 4 in znn.com.
2) The znnservice service (rainbow-znn’s server) and COMPAS J2EE framework are deployed on node 3, and Hyperic-Sigar is launched to collect the running information (e.g. CPU utilization, hard disk I/O, memory, network etc.).
3) The distributor-0.7 provided by install-lb.sh in the znn.com installation package is deployed on node 2 to realize the load balance of znndist service.
4) The znnclient service is deployed on node 1 to simulate the client of znn.com. In addition, Jmeter-2.5.1 is deployed on this node to simulate user access.

3.2. Experiment Process
1) Record environment information, here take CPU utilization as the targeted REE. The time-stamp and CPU utilization data are obtained, where the data set is obtained in increments of 5 minutes as shown in figure 3.
2) Perform differential processing on collected data and remove the periodicity and trend of the data. From figure 3, it can be seen that the trend of CPU utilization does not meet the need of stationarity, so the original time series are processed by the first-order difference. The data after the first-order difference is shown in figure 4, so the difference order d is set to 1.

![Figure 3. CPU utilization rate chart.](image)  ![Figure 4. Data after the first difference.](image)

3) Perform autocorrelation and partial autocorrelation analysis on the data after the first difference. A python script is given to calculate the values of ACF and PACF, as shown in figure 5 and figure 6.
Figure 5. ACF chart after the first difference.  

Figure 6. PACF chart after the first difference.

4) Estimate the model parameters and determine the order of \( p \) corresponding to ACF, the order of \( q \) corresponding to PACF. Here the order of \( p \) is 1 and the order of \( q \) is 4, and the model is shown in (3):

\[
y_t = 0.808 \times y_{t-1} + \varepsilon_t + 1.088 \times \varepsilon_{t-3} - 0.359 \times \varepsilon_{t-1} + 0.259 \times \varepsilon_{t-4}
\]

where \( y_t \) is the observed value of CPU utilization at time \( t \), and \( \{\varepsilon_t\} \) is the standard normal white noise sequence, \( E(\varepsilon_t) = 0, \text{Var}(\varepsilon_t) = \sigma^2, E(\varepsilon_t, \varepsilon_s) = 0, s \neq t \).

5) Significance test and residual analysis. The significance test results are shown in table 2.

Table 2. Significance test results.

| Model       | Model fitness statistics | Ljung-Box Q(18) |
|-------------|--------------------------|-----------------|
| CPU_B-model | Stationary               | R²              | Statistics   | DF | Significance |
|             | 0.153                    | 0.722           | 70.222       | 14 | 0.000        |

| CPU_B-model | CPU_B | No conversion | AR | Lag | Estimate | SE  | T   | Significance |
|-------------|-------|---------------|----|-----|----------|-----|-----|--------------|
|              |       |               |    |     | 1        |     | 22.667 | 0.000         |
|              |       |               |    |     | 1.088    | 0.35 | 30.661 | 0.000         |
|              |       |               |    |     | -0.359   | 0.44 | -8.111 | 0.000         |
|              |       |               |    |     | -0.259   | 0.37 | 7.007  | 0.000         |

Perform residual analysis on ARIMA(1,1,4) and the analysis results are shown in figure 7. Therefore, ARIMA(1,1,4) fits reasonably.

Figure 7. Analysis result of residual error.

6) Predict and analyze the trend of CPU utilization behavior. The corresponding analysis results are shown in figure 8. It can be seen that after the 185th data point, the CPU usage rate starts to rise and will continue to rise and exceed the threshold.
4. Conclusion
This paper proposes the behavior trend analysis method of REEs based on ARIMA, which can analyze and predict how the behavior of REEs will change. Through the historical data collection of REEs, analysis of their future behavior trends is realized. Thus before the actual changes of REEs happen, the corresponding adaptation strategies can be determined in advance according to the analysis results, or the corresponding adaptation strategies can be formulated online.

Acknowledgments
This work was supported in part by the Defense Key Laboratories Support Program of China (Grant No. WDZC2019601A303).

References
[1] Jiang HT, Liu Y, Lin HS, et al. 2016 Environment model of self-adaptive software and its implementation Journal of Frontiers of Computer Science & Technology vol 10 no 3 pp301-310 (in Chinese)
[2] Cicirelli F, Giordano A and Nigro L 2015 Efficient environment management for distributed simulation of large-scale situated multi-agent systems Concurrency and Computation: Practice and Experience vol 27 no 3 pp 610-632
[3] Yang ZQ and Jin Z 2018 Self-Adaptive Decision Making Under Uncertainty in Environment and Requirements Journal of computer research and development vol 55 no 5 pp 1014-1033 (in Chinese)
[4] Calinescu R, Weyns D, Gerasimou S, et al 2017 Engineering Trustworthy Self-Adaptive Software with Dynamic Assurance Cases IEEE TRANSACTIONS ON SOFTWARE ENGINEERING pp 1-1
[5] Alferez G H, Pelechano V, Mazo R, et al 2014 Dynamic adaptation of service compositions with variability models Journal of Systems & Software vol 91 no 5 pp 24-47
[6] Cai L, Jiang F, Xu WX, et al 2019 Research on a Clustering Algorithm Based on Adaptive Grid Partition and Decision Graph Journal of Chinese Computer Systems vol 40 no 10 pp 2033-2038
[7] Becker 2015 A survey on engineering approaches for self-adaptive systems Pervasive Mobile Computing vol 17 pp 184-206
[8] Zhao TQ, Zhao HY, Zhang W, et al 2018 Survey of model-based self-adaptation method Journal of software vol 29 no 1 pp 23-41 (in Chinese)
[9] Garima J and Bhawna M 2017 A Study of Time Series Models ARIMA and ETS SSRN Electronic Journal pp 1-1
[10] Cheng SW, Garlan D, Chair J, et al 2016 Rainbow: Cost-Effective Software Architecture-Based Self-Adaptation Thesis Committee Carnegie Mellon University (Pittsburgh: Schenley Park)