A Verification Technique for Self-Adaptive Software by Using Model-Checking

Euijong Lee[1] and Doo-Kwon Baik[2]
Department of Computer and Radio Communications Engineering,
Korea University
Seoul, Republic of Korea
{kongjjagae[1], baikdk[2]}@korea.ac.kr

***Abstract*— Self-adaptive software refers to software that can change its behavior by itself to perform an intended objective according to changes in the surrounding environment. In this study, a technique is proposed to detect the possibility of problem occurrence in advance, before the self-adaptive software carries out self-adaptation according to changes in the surrounding environment. For the proposed technique, a model-checking technique is applied, which is a software modeling verification technique. The proposed technique was applied to ZNN.com, which provides a self-adaptive software scenario. Through this, the possibility was shown that an adaptive-strategy can be prepared in advance by self-adaptive software through the model-checking technique in the stage prior to the execution of self-adaptation.**

Keywords— Self-adaptive software, Model-Checking, Software verification

I. Study Background

For conventional software, a problem is defined and the method, according to which it is solved, is constructed at the development stage. With these types of software development methods, when a problem that did not exist at the development stage occurs, a software error occurs or in a serious case, the software becomes inoperable. To solve this problem, the concept of self-adaptive software has emerged. Self-adaptive software refers to software that changes by itself according to environmental changes, thereby solving a problem. Currently, studies are being carried out on self-adaptive software in various aspects in different fields. This study was conducted for problem detection, which is one of the research areas for self-adaptive software, and it aims to detect and prevent the possibility of problem occurrence in advance before a change of environment occurs.

II. Proposed Method

This study was carried out to identify a software problem before it happens by using the model-checking technique, and to prevent it through self-adaptation on the basis thereof. The proposed method was based on MAPE-K proposed by IBM. Because this study aims to detect a problem in advance in a stage before it happens, the model-checking part was added and subdivided in the monitoring stage, and each stage of MAPE-K, except for monitoring, performs the inherent roles.

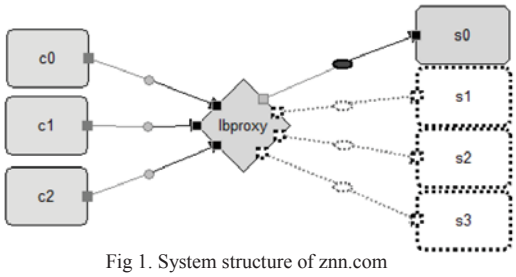

Fig 1. System structure of znn.com

III. Case Study

A. Scenario: ZNN.com

ZNN.com is a case study used by Rainbow to evaluate self-adaptive software, and it consists of the structure shown in Fig. 1 [1]. Znn.com is a virtual news media website. Znn.com has the goal of providing a reasonable response time to a user. Znn.com provides three service stages to its users according to the server load. For example, when a server overloads, only texts are provided, and when a server has available capacity, articles are provided along with various media by improving the quality of service. Furthermore, when the data traffic increases considerably, a strategy is adopted to reduce the time required to respond to users by increasing the number of operable servers. Consequently, two criteria can be selected to execute self-adaptation for znn.com. The first one is performance, which can be the response time, the server load, and the network bandwidth. The second one is cost, which can be expressed by the number of operating servers.

B. Application of Case Study

An experiment was carried out to show that the possibility of problem occurrence can be identified in advance by using the model-checking technique. The znn.com scenario mentioned in 3.A was used as the scenario for the experiment. In accordance with the scenario, the server loads were defined in Table 1 according to the service types, and the amount that can be processed by the server was defined as 300 M/s. In accordance with the experimental scenario of znn.com, the environmental variables that will be used by the proposed technique before defining the FSM were defined in Table 2.

TABLE 1. TRAFFIC ACCORDING TO SERVICE

Quality of Service	Used Traffic
High	3 MB/S
Medium	2 MB/s
Low	1 MB/s

TABLE 2. THE ENVIRONMENTAL VARIABLES USED IN THE SCENARIO

Environment Variable	Description
User number	The number of users accessing the ZNN.com service
Service quality	It means the quality of service and it is 3 in the case of highest service quality, 2 in the case of medium service quality, and 1 in the case of the lowest service quality
Running server number	The number of servers currently operating

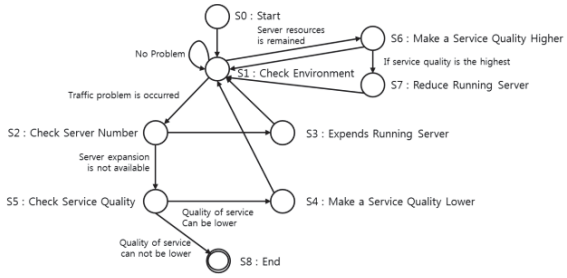

Figure 2. FSM representing znn.com

Based on the above assumption, the FSM that was constructed to use the model-verification technique for verification is shown in Fig. 3. S0 is the stage for starting the model verification. S1 is the stage for confirming the current status through the environmental variables. In the pertinent model, the service status was identified with the following equation.

$$S(e) = \frac{Service\ quality\ \times User\ Number}{Running\ Server\ Number\ \times Capacity\ of\ Server}$$

If the value of S(e) is larger than 1, it is determined that there will be no problem in providing the service, and if smaller than 1, it is determined that there will be a problem in providing the service. S2 is the stage for confirming the available servers in addition to the currently operating servers, and when there are available servers, S3 is the state in which they are used. When there is no available server, S4 is the stage for checking the service quality to determine if it can be reduced, and if there is a level to which the service can be lowered, the service quality is decreased through the S5 stage. S6 and S7 means that if server resource remain, make better quality or reduce running server to reduce cost. Lastly, S8 means that the service is terminated because there are no further possible methods. Furthermore, the case of not satisfying the S1 stage means that self-adaptation is required, and the S3 and S5 stages mean that self-adaptation has been applied.

For the proposed FSM model, the number of users was defined as 0–10,000 by using a NuSMV model-verifier; and the values were checked for the environmental variables occurring in S3 and S5 where self-adaptation is required, and S6 where the service operation is impossible. The pertinent results are shown in Table 3.

TABLE 3. VALUES OF ENVIRONMENTAL VARIABLES THAT REQUIRE SELF-ADAPTATION

Environmental Variable Value	Status	Adaptation Strategy
User number: 101/201 Service quality: 3/3 Running server number: 1/2	S3	Increase a running server
User number: 301 / 451 Service quality: 3 / 2 Running server number: 3 /3	S5	Decrease the service quality
User number: 99 / 199 Service quality: 3 / 3 Running server number: 2 /3	S7	Decrease a running server
User number: 299 / 449 Service quality: 2 / 1 Running server number: 3 /3	S6	Increase the service quality
User number: 901 Service quality: 1 Running server number: 3	S5	Impossible to provide the service

IV. CONCLUSION

This study was carried out to extract the adaptation stages of self-adaptive software and prepare for them in advance. For this, a technique was proposed for extracting the environmental variables, which are required for self-verification in the monitoring stage during the MAPE-K process used in conventional self-adaptive software, and based on this, extracting the self-adaptive stage in advance through model-checking. The proposed technique was applied to the znn.com scenario, which is used as a case study in self-adaptive software studies. Through the applied results, the surrounding environment where self-adaptation is required was extracted prior to executing an adaptation-strategy by using the model-checking technique.

In future, a follow-up study will be carried out to use the proposed technique during actual execution time. Furthermore, a study will be carried out to develop a technique that extracts or generates an adaptation strategy through model-checking.

ACKNOWLEDGMENT

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF 2012M3C4A7033346). Doo-Kwon Baik is corresponding author.

REFERENCES

[1] Cheng, Shang-Wen, David Garlan, and Bradley Schmerl. "Evaluating the effectiveness of the rainbow self-adaptive system." Software Engineering for Adaptive and Self-Managing Systems, 2009. SEAMS'09. ICSE Workshop on. IEEE, 2009.

[2] Mazeiar Salehie and Landan Tahvildari, "Self-Adaptive Software: Landscape and Research Challenges", In TAAS, Vol 4, Issue 2, pp14-42, May 2009.

[3] Biere, Armin, et al. Symbolic model checking without BDDs. Springer Berlin Heidelberg, 1999.

[4] Baier, Christel, and Joost-Pieter Katoen. Principles of model checking. Vol. 26202649. Cambridge: MIT press, 2008.

[5] Garlan, David, et al. "Rainbow: Architecture-based self-adaptation with reusable infrastructure." Computer 37.10 (2004): 46-54.

[6] Principles of model checking. Vol. 26202649. Cambridge: MIT press, 2008.