Detecting Errors and Reusing Resource in Self-Adaptive Module*

Joonhoon Lee, Jeongmin Park, Eunseok Lee

Department of Electrical and Computer Engineering, Sungkyunkwan University,
Suwon 440-746, South Korea
{trsprs, jmpark, eslee}@ece.skku.ac.kr

Abstract. Computing environments have been increasingly complex, making managing them difficult. Self-Adaptive methods are used because these do not reveal internal structure and adapt to the environment. Our previous study uses an activation switch that determines components' tasks. However, they may not work due to the external influences such as viruses because test cases do not contain all cases. If many processes are executing, the shut-down component can waste system resources. In this paper, we propose detecting errors that occur in the Self-Adaptive module and reuse the resources that the shut-down component locks. This can reduce the number of active component when many processes have executed and make the system safer. The paper applies the proposed approach to a video conferencing system to evaluate it. This compares the adaptation of the several approaches with this approach. When many file transfer tasks are active, the number of components is compared. In these experiments, we found abnormal states from the previous study and reduced resource usage through resource reuse.

Keywords: Self-Adaptive, Self-Healing, Activation Switch, Monitoring, Component

1 Introduction

Computing environments have been increasingly complex. Some problems occurred in complex system are difficult to analyze and resolve [1], [2]. One approach is managing the system directly. For example, if the errors are occurred, the system manager can execute a new application or change the configuration of a server to fix the error. However, this approach often requires complex steps and depends on the manager's capability [3]. Self-Adaptive systems can diagnose system errors, handle a variety of resources, and manage the user requirement. Such a solution can reduce the spiraling costs of managing component systems.

* This work was supported by the MKE 21st Century Frontier R&D Program in Korea and a result of subproject UCN 08B3-B1-10M, ITRC IITA-2008-(C1090-080-0046), Grant No. R01-2006-000-10954-0, Basic Research Program of the Korea Science & Engineering Foundation.

The Self-Adaptive module can adapt the target system, but the module itself can still have another fault. Our previous study [4] proposed an activation switch. This reduces resource waste caused by the use of unneeded components and improves the system performance. However, software testing always does not consider all of the cases. Problems that have never been expected in development phase or testing phase are sometimes occurred. The Self-Adaptive system can consider these situations. In this paper, we assume the activation switch cannot work normally when the system has external problems, such as a virus that prevents change to the component state. This makes the Self-Adaptive module abnormal and thus the system cannot provide the adaptation service. It also uses much memory space with as many tasks as ever.

This paper proposes the following solutions to these problems: 1) Check the activation switch and recover the abnormal switch and component; 2) Reuse the component and resource not used in the current level. These reduce the number of components with many tasks and prevent resource waste. The Self-Adaptive module can operate with more safety. In this paper, we implemented a simple video conference system to evaluate the proposed approach. The experiment evaluates this approach, resolves the state that the previous study does not resolve. It also compares the number of components before and after applying the approach. These confirm the Self-Adaptive module's safer condition.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 describes our proposal. Section 4 describes implementation and evaluation of our proposal. Finally, we discuss the conclusion and future work in Section 5.

2 Related Work

This section discusses our previous study [4] and reviews Stelling's Heartbeat Monitoring [5]. We also discuss GridFTP [6] and File Replication Monitor [7], [8].

Table 1. Four levels of Self-Adaptive File Transfer

Level	Description
Judging address	Checking whether address is approved
Estimating time	Estimating file transfer time
Transferring files	Monitoring file transfer
Self-Adaptation	Blocking a suspicious file

Our previous study proposed a Self-Adaptive file transfer system [4]. This system monitors the file transfer state. If suspicious behavior such as a virus is detected, the system reports a warning message to the user or the manager. They can determine the files deleted and fix them and/or block the sender's IP. In this process, it categorizes the function level, shown in Table 1. Some switches point to each level. If the function of each level should be executed, the switch is turned on and the components of that level are executed. Other components are turned off. This means the activation switch prevents all components from run continuously and prevents resource waste in the target system.

This switch may be stopped by an external behavior such as a virus that infects the memory in the system, so it cannot change the state of the component. The activation switch does not have a counter plan. It also has a disadvantage in that the number of executing components is increasing when many tasks occur. This decreases system performance.

Stelling [5] proposed a monitoring approach that has a periodic state check. A component sends a periodic signal, "I am alive", and the monitor receives the signal and determines that the component is in the normal state. If the signal does not come within a certain time, the monitor determines the component is in an abnormal state. The advantage of this approach is that the monitor does not check the state of the component directly. However, if the periodic signal is delayed or the monitoring period is too short, the monitor has greater receiving signal overload.

GridFTP [6] provides a high-performance, secure, reliable data transfer using a protocol optimized for high-bandwidth wide-area network based on FTP. This also provides automatic backup. Though this can have high performance and reliable capabilities, adaptation of current state does not be supported. For example, when an invalid user intrudes into a transfer, this continues the transfer without validation. When same situation is occurred, this behaves same way. If the sender is illegal user, the transfer is started once. So this does not provide Self-Adaptation ability.

File Replication Monitor [7], [8] provides real-time automatic backup, replication and synchronization for local and remote files and directories. It can monitor file size, free space, extensions and so on. But this does not provide any adaptation. If some problems are occurred and this does not have any solution, this should not provide functional operations.

This paper describes the problem based on related work: 1) Self-Adaptive module does not have a counter plan when an external factor makes the Self-Adaptive component abnormal; 2) Resources are wasted when many tasks are executed.

3. Proposed Approach

This paper proposes detecting errors in the activation switch using the heartbeat monitoring technique [5]. Figure 1 shows the proposed architecture. The left side depicts a Self-Adaptive target module. This module monitors the target system. If the target system has suspicious behavior, the module adapts the target system to a safe state. The right side shows the proposed architecture. This has the following objectives: 1) Detection of abnormal state of the Self-Adaptive components and healing them; 2) Executing alternative components to replace the Self-Adaptive component; 3) Reusing some components unused at this level. In this case, these components mean the Self-Adaptive components. Details of proposed components are following.

- State Monitor: In [4], Park determines the state of activation switch of file transfer components. State Monitor monitors which state is current state. This monitoring can know components that should be worked in this time.
- Configuration Monitor (CM): The architecture of Safe File Transfer System has four layers for Self-Adaptation (Probes layer, Gauges layer, Controllers layer,

Effectors layer). Probes layer monitors the target system. These monitored values are transported to Gauge layer by the Probe buses. Gauge layer evaluates those values. The evaluated factors go to the Controllers layer.

- Configuration Repository (CR): It saves the values from Configuration Monitor. During file transferring, it can provide information of component that is in Self-Healing state.
- Signal Monitor: When a Self-Adaptive component becomes active state, it sends the starting signal to Signal Monitor. Signal Monitor monitors these signals. Periodical monitoring method may have large monitoring workload [5]. Signal Monitor does not monitor the target periodically. Instead of that, the target sends its current state to the monitor.

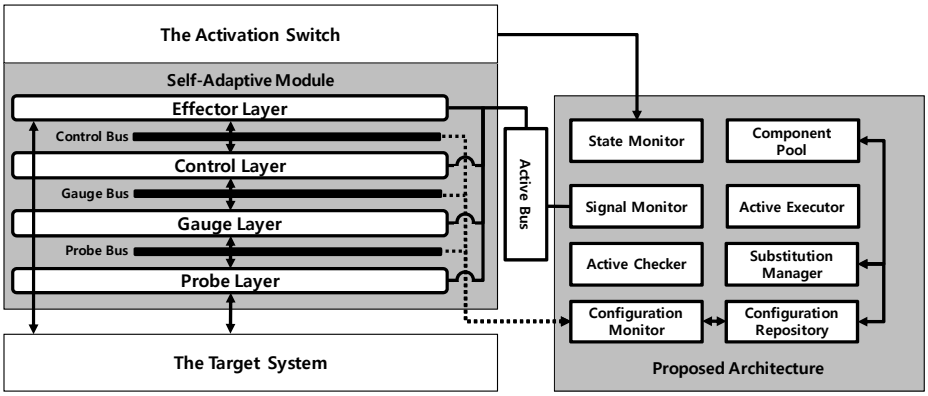

Fig. 1. Proposed Architecture on right side. Left side depicts the target Self-Adaptive module and the target system.

- Active Checker: When the activation switch is on-state, component related to that switch must be active state. Active Checker (AC) checks the state of component that may have error occurred by activation switch. If AC judge components related to the active switch do not work, these components are considered as error-state and AC notifies this knowledge to Active Executor.
- Active Executor: Active Executor (AE) applies the Self-Healing strategy to the component AC notifies. In this paper, the strategy is to quit that component and start new component. This new component is provided by Component Pool.
- Substitution Manager: Components that are healing by itself do not provide their functional behaviors [3]. This component creates the alternative components of each level (See Table 1). To solve this problem, Substitution Manager receives switch information and generates the alternative components of next level in advance. These components are saved in Component Pool. If the type of alternative component is already saved in Component Pool, Substitution Manager does not generate new alternative component. The alternative components can be used for reusing process. If one task A is processing, a task B can use components that are not currently used by task A. This reuse process describes in 3.2.
- Component Pool: It saves the alternative components generated by Substitution Manager and provides them if needed.

3.1 Error Detection Process

An error means the activation switch works incorrectly or the state of component and the switch are unequal. The switch represents the current level of the target Self-Adaptive module. If it is incorrect, the module does not work as well. So, the target system does not receive a Self-Adaptive service. The process steps are follows.

- Recognizing current State (Step 1): The Safe File Transfer System includes four transfer levels: 1) Address Approval Judgment; 2) Estimates file transfer time; 3) File transfer; and 4) Self-Adaptation. State Monitor monitors current file transfer level. This lets the proposed architecture know the current level of the Self-Adaptive module. Through this, it correctly knows the current level. If the current level progresses to the next level – for example, from Address Approval Judgment to Estimate file transfer time, it goes to the next step.
- Generating alternative component (Step 2): The current working components are based on the activation switch. This step generates the same type components of the current working components. If the Component Pool already has the same component, the component does not generate the alternative.
- Monitoring signals (Step 3): When the components in the Self-Adaptive module start to work based on the activation switch, the components send the active signal to the Active Bus. Signal Monitor checks the Active Bus when the states' activation switches are changed. If there is no signal component, Signal Monitor notifies the Active Checker.
- Checking the behavior state of the component (Step 4): This step checks the behavior state of the component that is reported by the Signal Monitor. Active Checker checks the behavior state. As the activation switch is on-state, the related component must be working by the activation switch rule (See Table 1). If the component does not work, it notifies the component to the Active Executor.
- Applying the Self-Healing Strategy (Step 5): In this step, the component error is self-healed. The Self-Healing process does not provide component functionality [1]. Substitution Manager fetches the alternative component to replace the healed component from the Component Pool. This alternative requires the current state of healed component. Configuration Monitor monitors four layers (Probes, Gauges, Controllers, Effectors) through the active bus. These are saved in the Configuration Repository. The alternative component obtains information from this repository and uses it to function alternatively. The available Self-Healing strategies are rebooting, reconfiguration, and alternation. This paper uses the alternative method.

3.2 The process of reusing Self-Adaptive component

The activation switch classifies the process levels and follows the order of these levels like a hardware pipe line. If the level is four and a process A is on the third level and the other process B is on the second level, B can use A's component unused in this level. The process steps are follows.

- Recognizing current State (Step 1): When a new process is started or the level switches to the next level, the State Monitor checks other activation switches and finds up-level switches. For example, if one activation switch is level 2, the State

Monitor finds 3, 4, or higher level activation switches. This information is used by the next step.

- Saving component information (Step 2): The Configuration Monitor saves the information, such as parameters or transfer values in the Configuration Repository. This information is used by the Substitution Manager.
- Reusing component (Step 3): In this step, the Substitution Manager uses information from the Configuration Repository and inputs this to the alternative component. Finally, this manager replaces the component by this alternative component.

4. Implementation and Evaluation

We selected a target system and a Self-Adaptive module. The target system is a video conferencing system. The Self-Adaptive module is the safe file transfer module. It provides suspicious behavior detection and self-adaptation. The target system, a video conferencing system, has the objective to conduct a successful meeting. During conferencing, the users can send and receive documents. In this process, we consider many file transfer tasks.

We implemented the target system and the Self-Adaptive module. The server on the target system and the Self-Adaptive module were implemented in Java. The client and monitoring tools are implemented in C#. We used DirectShow for imaging processing.

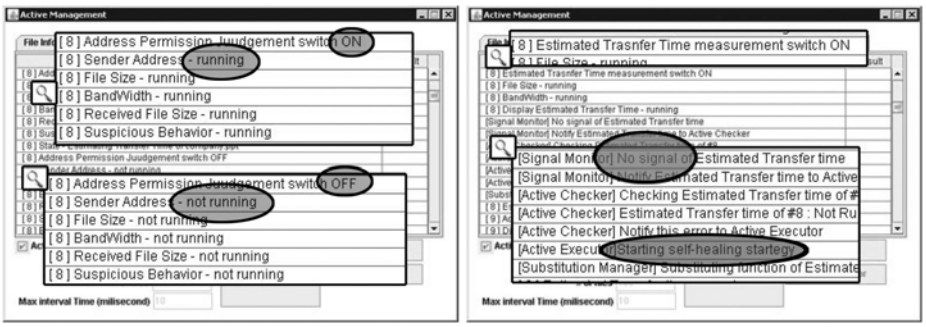

Fig. 2. Normal State and abnormal state of Activation Switch

The normal and abnormal state file transfer processes are shown in Figure 2. This figure shows the logs of the state of components. If the activation switch is on-state, the related components start to run. If the activation switch is off-state, the related components turn off (Circles on left window in Figure 2).

We arbitrarily forced the Self-Adaptive module to be in an abnormal state with an error rate 0.01. Right window in Figure 2 shows the estimate level switch is on, but the related components not running (Circles on right window in Figure 2). The Signal Monitor detected the components, because the signal did not arrive at the Signal Monitor. After detection, the Active Checker checks if the components were really

not running. If not running, the Active Executor applies the healing strategy to the related component.

4.3 Evaluation

We implemented the video conferencing system and safe file transfer module to apply our approach. In this section, we compare the previous safe file transfer module with the proposed applied module. Next, we compare the number of components in the two cases, before and after applying the proposal. If proposed approach is not applied, many tasks are not considered. But proposed approach can consider component reuse and reduce the usage of the resource. This also can detect errors occurred by the activation switch and heal it by itself. Besides, this retains the component function during self-healing state.

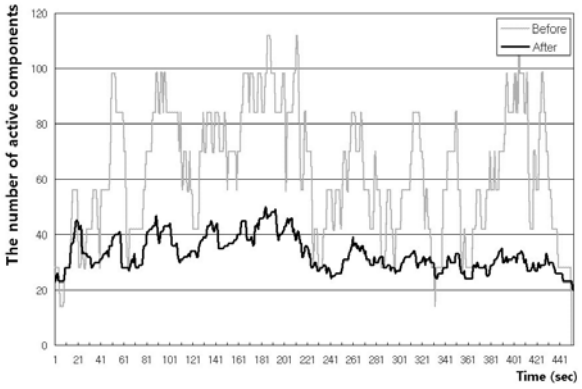

Fig. 3. The number of active components before applying the proposal and after applying it. Due to component reuse, the proposal reduces the number of active components.

Table 2. Comparison with other approaches

Feature	GridFTP	Replication Monitor	Our Proposal
Self-Adaptation	-	-	✓✓
Reliable File Transfer	✓✓	-	✓
Considering current state	✓	✓	✓✓
Reducing Monitoring Workload	-	✓	✓
External Error Monitoring	✓	✓	✓✓
Performance Improving	✓	✓	✓

- Not consider ✓ Weak advantage ✓✓ Strong advantage

We also executed many tasks to evaluate the reuse process. In this paper, we transferred 100 files; each file transfer interval is randomly between 0 and 1 second. Figure 3 shows the experimental results. During the process, the number of components when the proposal is applied is lower than without it. That is, the system resource usage is reduced.

Table 2 shows comparison of these approaches. This shows our proposal can provide better Self-Healing abilities than other approaches.

5. Conclusion & Future work

This study proposed an error detection method in the Self-Adaptive module and the unused component reuse method in the current state when many tasks are active. These are the advantages of the proposed approach. First, it makes the Self-Adaptive module safer. Second, it reduces the number of current active components and improves system performance.

However, when the module that heals the Self-Adaptive module such as our approach has a problem, another approach is needed. This may increase system overload. This case may be reported to the system manager.

Future approaches to this research will consider state diagram representation to show the current system state and where the problem occurred. And the proposal in this paper inserts the knowledge of total system manually. We will also propose the representation method of this knowledge.

References

1. Jiwen Wang, Chenghao Guo and Fengyu Liu: Self-Healing Based Software Architecture Modeling and Analysis Through a Case Study. In: 2005 IEEE Networking, Sensing and Control, pp. 873—877. Arizona (2005)
2. David S. Wile and Alexander Egyed: An Externalized Infrastructure for Self-Healing Systems. In: the 4th Working IEEE/IFIP Conference on Software Architecture, pp. 285—288. Oslo, Norway (2004)
3. Robert Laddaga: Creating robust software through self-adaptation. Intelligent Systems and Their Applications, vol. 14, pp. 26—29. IEEE Computer Society (1999)
4. Jeongmin Park, Joonhoon Lee, Eunseok Lee: Goal graph based Performance improvement for Self-Adaptive Modules. In: ACM SIGKDD International Conference on Ubiquitous Information Management and Communication (ACM SIGKDD ICUIMC2008), pp. 68—72. Suwon, Korea (2008)
5. Paul Stelling, Ian Foster, Carl Kesselman, Craig Lee, Gregor Von Laszewski: A fault detection service for wide area distributed computations. Cluster Computing, vol. 2, pp. 117—128. Kluwer Academic Publishers (1998)
6. William Allcock, John Bresnahan, Rajkumar Kettimuthu, Michael Link: The Globus Striped GridFTP Framework and Server. In: the 2005 ACM/IEEE conference on Supercomputing, pp. 1—11. Seattle (2005)
7. Anupam Bhide, Elmootazbellah N. Elnozahy, Stephen P. Morgan: Implicit Replication in a Network File Server. In: Management of Replicated Data, pp. 85-90. Houston, USA (1990)
8. File Replication Monitor, http://www.file-monitor.com/file-replication-monitor.html