기능 기반 시스템의 모니터링을 위한 방법론

유보식, 유길종, 정진수, 이은석
성균관대학교 컴퓨터공학과
e-mail : well1211@hotmail.com, {nglover, seba702, eslee}@ece.skku.ac.kr

A Study on Monitoring Function Based System*

Bosik Ryu, Giljong Yoo, Jinsu Jung, Eunseok Lee
Dept. of Computer Engineering, Sungkyunkwan University

요 약

근래에 들어와 시스템의 복잡성에 대한 문제가 제기되고 있다. 그 해결책으로 대두되고 있는 자가치유 시스템이란 자율 컴퓨팅의 개념 중 하나로 사람의 개입 없이 시스템의 이상상태를 인식하고 정상상태로 복귀 가능한 시스템을 의미한다. 이를 달성하기 위해서는 가장 먼저 필요한 것은 시스템의 어떤 부분에서 어떤 이상이 일어났는지 파악하는 것이다. 그래서 내외부적으로 여러 가지 접근을 통한 시스템에서의 이상여부를 파악하기 위한 모니터링 기법들이 연구되고 있다. 그렇지만, 기존의 여러 모니터링 기법에서는 부분에 초점을 두어 전체적인 시스템의 감시를 위한 다른 방법이 필요하다.

본 논문에서는 이를 해결하기 위하여 현재 부곽되고 있는 AOP(Aspect Oriented Programming)기법을 적용하여 각 기능의 수행범위 / 제약사항을 조절하며 기능적인 작은 이상을 파악한다. 또한, 모듈-기능 관계리스트를 사용하여 이상이 발생하였을 경우 어떤 요구사항이 위반되었는지 파악하고, 해당 기능들에 근거한 State-diagram을 사용함으로 시스템의 전체 흐름 상의 이상 발생을 감시하는 구조를 제안한다. 그리고 Case Study를 통해 실제 이 방법을 적용한 시스템의 감시가 가능하다는 것을 증명했다.

1. 서 론

근래에 들어 시스템의 복잡성이 발생시키는 문제점들을 해결하기 위한 방법으로 자율 컴퓨팅이 주목받고 있다. 자율 컴퓨팅(Autonomic Computing)이란 자가구성(Self-Configuring), 자가치유(Self-Healing), 자가최적화(Self-Optimization) 그리고 자가보호(Self-Protecting)의 네 가지 특성을 가지고 있는 시스템이다. 이 네 가지 특성은 지속적으로 변하는 환경, 사용자들의 요구, 그리고 발생하는 이상(Fault)들을 다룰 수 있는 능력을 의미한다.[1]

본 논문에서는 자율 컴퓨팅의 개념 중 자가치유에 대하여 다루고자 한다. 자가치유 시스템은 현재 활발한 연구가 일어나고 있는 분야로 이는 사람의 개입 없이 시스템의 이상상태를 인식하고 정상상태로 복귀 가능한 시스템으로 세가지 주요 기능으로는 다음과 같다.

첫 번째로 '건강한 상태 지속(Maintenance of health)'과 두 번째로 '시스템 복구(System recovery)', 그리고 '시스템 이상의 발견(Detection of System failure)'이다.[2]

이 세 가지 역할 중 가장 우선시되어야할 역할은 '시스템 이상의 발견'에 있다. 시스템이 자신의 상태가 이상이 있는 지 없는 지 발견을 할 수 있어야 어떤 행동을 취할 것인지 시스템을 중지할 것인지 대한 결정을 내릴 수 있기 때문이다. 그러나 2장에서 살펴볼 기존의 연구들이 공통적으로 부분적인 관점에 중점을 두어 전체적인 관점의 감시가 소홀한 것을 발견할 수 있었다. 그래서 본 논문에서는 보다 광범위한 감시 방법을 제안한다.

제안 방법은 현재 부곽되고 있는 기술 중 하나인 AOP 기법을 사용한 모니터링 기법이다. 기본 구조는 시스템을 기능 단위로 나누어 시스템의 작은 기능적 이상부터 의도되지 않은 시스템의 흐름까지 파악하는 것이다. 또한, 궁극적으로 해당 이상을 사용자가 지정해 놓은 요구사항과 연결시켜 어떤 요구사항이 영향을 받았는지까지 파악할 수 있는 구조를 제안하고자 한다. 제안된 시스템의 가용성에 대해서는 메신저 시스템을 사용한 Case study를 통하여 검증했다.

본 논문의 구성은 다음과 같다. 2장에서는 기본 배경 지식 및 대표적인 관련연구들에 대하여 간략하게 기술하며, 3장에서는 제안하고자 하는 구조에 대한 자세하게 설명하고, 4장에서는 Case study를 통하여 3장에 기술된 내용을 검증하였으며, 5장에서는 결론과 향후 과제를 기술한다.

2. 관련연구

시스템 감시에 관련된 연구에서 많은 연구가 진행되었지만 크게 내부적 방식과 외부적 방식으로 나뉜다.

내부적 방식이란 말 그대로 프로그래밍 언어를 사용하여 해당 시스템 안에 감시하고자 하는 것에 대하여 구현해 놓은 것이다. 이에 반하여 외부적 방식은 감시 역할에 대한 부분을 다른 모듈에 위치시켜 다른 시스템들에 대한 분석, 수정, 확장, 재사용을 용이케 한 방식이다.[3][8]

내부적인 방식 중 대표적인 것은 Michael E. Shin의 자가치유 컴포넌트(Self-Healing Components) [4]이다. 이 방식은 각각의 Component마다 서비스 계층(Service layer)과 치유 계층(Healing layer)을 나눠 자가치유를 구현한다. 그렇지만, 해당 방식으로는 내부적인 구성요소에

* 본 연구는 지식경제부의 유비쿼터스컴퓨팅 및 네트워크원천기반기술개발사업, 교육과학기술부의 특정기초연구사업R01-2006-000-10954-0, 교육인적자원부의 2단계 BK21사업의 연구결과로 수행되었음

초점이 고정되어 있기에 시스템의 전체적은 흐름에 대한 감시 / 조정을 할 수 없다. 그래서 각각의 Component는 이상 없이 작동이 되어도 전체 시스템은 사용자의 의도와는 벗어난 행동을 할 수가 있기에 전체적은 시스템을 조정하기 위한 방법이 필요하다.

AOP 방법을 모니터링 기법에 적용한 연구로는 Martin Gogolla의 Aspect-Oriented Monitoring of UML and OCL Constraints[5]가 있다. 이는 UML과 OCL(Object Constraint Language)을 AOP 기법에 적용시켜 시스템 모델링 시 제공된 제약사항이 유지되는 지 감시하는 방식이다. 그렇지만 이 연구 역시 어떤 pointcut 부분에 해당하는 제약사항들이 유지되는 가에 중점을 두었다. 그렇기에 해당 관점(Aspect)은 이상이 없이 지나가도 전체적인 시스템의 흐름에는 이상이 발생할 수 있기에 조금 더 광범위한 모니터링 방법이 필요하다.

또한 William N. Robinson의 Requirements monitoring framework[6][10]는 사용자의 요구사항이 위반되었는지 파악하기 위한 구조이다. 그렇지만 여기서 제시된 ReqMon을 사용하기 위해서는 복잡한 KAOS(Knowledge Acquisition in Automated Specification) 언어로 상세화하여야 하기에 실질적인 사용이 제한된다.

따라서 본 논문에서는 위와 같은 문제들을 해결하고자 자가치유의 역할 중 '시스템 이상의 발견'에 중점을 두고 기능을 기반으로 시스템을 구성되었을 경우 해당 기능 각각의 감시부터 시스템의 전체적인 흐름에 대한 감시하면서 동시에 간단하게 사용자의 요구사항의 위반에 대해서도 감시할 수 있는 방법을 제시한다.

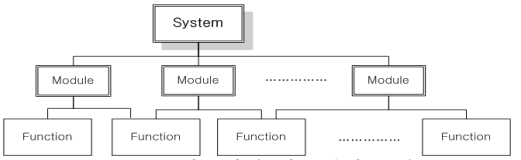
3. 제안 방법

제안되는 구조에서 사용되는 시스템의 아키텍쳐는 기능(Function)과 모듈(Module)로 구성한다. 기능이란 프로그래밍 시 생성된 시스템이 실행을 하게 되는 최소 단위의 함수를 의미하고. 모듈이란 시스템에서 어떤 한 가지 고유의 역할을 담당하고 있는 Agent를 의미한다. 모듈은 여러 개의 기능을 실행하여 결과를 받아 반영하며, 시스템은 이러한 모듈들이 서로간의 상호작용을 통하여 이루어지는 것을 의미한다.

그래서 전체적인 시스템의 구조를 기능 단위로 나누게 된다면 그림1과 같다. 가장 하위에는 기능을 가지고 그 기능들을 서로 공유하거나 혹은 배타적으로 사용하는 모듈들을 가지고 있다.

여기서 각 기능은 상황에 따라 다양한 입력값과 주어진 변수들을 가지고 실행 된 후 결과값 생성 및 변수들을 변환한다. 즉 어떤 기능을 실행하든 주어진 입력값 / 변수들과 결과값 / 변환된 변수들의 체크를 통해 해당 기능 내에서의 이상여부의 파악이 가능하다.

이를 확장하여 해당 기능에서 어떤 이상이 일어났는지 파악하여 이에 대한 영향을 파악 가능하다면 부분적인 감시 뿐 아니라 전체적인 시스템의 감시가 가능할 것이라는



(그림 1) 기본적인 시스템의 구성

것이 제안의 기본 내용이다.

이 제안 형태를 그림 1과 같은 그림으로 나타내면 그림 2와 같이 나타낼 수 있다

조작자(Controller)가 포함하는 자료들에 대해 정의하자면 다음과 같다.

기능-모듈 관계 리스트란 간단한 요구사항의 위반을 파악하기 위해 사용된다. 이 리스트는 어떤 모듈에서 어떤 기능을 사용하였을 때 이상 발생여부에 따라 영향을 받는 요구사항들을 포함한다. 따라서 이상 발생 시 해당 리스트를 사용자에게 보여주어 어떤 요구사항의 위반이 일어났는지 파악할 수 있도록 한다.

기능 기반 상태 다이어그램(Function based State diagram)은 전체 흐름의 상태도를 기능을 기반으로 작성된다. 이를 통해 현재 시스템이 어떤 상태에 있는지, 변화(Transition) 발생 시 해당 사항이 전체 흐름에 있어서 일치하는지 파악이 가능하다.

기능들의 수행범위 / 제약사항은 어떤 기능이 실행될 경우 기능 감시자(Function Monitor)에게 해당 입력값 / 결과값 및 관련 변수들에서 이상이 일어났는지 파악하기 위한 기준이 된다.

각 각의 역할에 대해 정의하면 다음과 같다.

기능 감시자는 기능의 수행 전

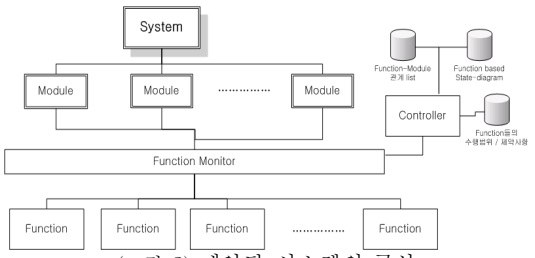
1) 조작자에게 사용되는 모듈-기능을 보고
2) 조작자로부터 제약사항 및 수행범위을 수신
3) 입력값 및 관련 변수들의 이상여부 확인 및 보고

기능의 수행 후

1) 결과값 및 관련 변수들이 제약사항에 맞는지 확인
2) 조작자에게 기능의 수행 완료 및 이상여부를 보고

조작자는 기능-모듈 관계 리스트, 기능 기반 상태 다이어그램 및 기능들의 수행범위 / 제약사항을 보유하고 있으며 기능의 수행 전

1) 기능 감시자로에게 제약사항 및 수행범위를 전송
2) 기능 감시자로부터 보고된 상황에 기반한 상태 다



(그림 2) 제안된 시스템의 구성

이어그램을 체크하고 이에 따른 이상여부 확인

기능의 수행 후

1) 기능 감시자로부터 보고된 기능 완료 후의 이상여부에 따른 상태 다이어그램 내 이상여부 확인

오류 발생 시

1) 실행된 기능과 모듈의 관계에 관련된 요구사항 리스트에서 해당되는 위반 사항을 파악

4. 평가-Case Study

그림 3와 같은 메신저 시스템은 총 네 개의 모듈들을 가지고 있고 총 일곱 개의 기능을 가지고 있다.

현 메신저 시스템의 기본 기능은 다음과 같다.

1) Message 송수신

2) Message 송/수신의 기록

또한, 기본 요구사항으로는 간단히 다음과 같다

1) 메시지 송신 시 3초 이내에 전송되어야 한다.

2) 인증된 사용자만 메시지 송수신이 가능해야 한다.

3) 사용자마다 자신의 주소록을 가진다.

4) 사용자는 메시지 송수신 / 송수신내용 기록 / 주소록 확인이 가능하여야 한다.

5) 누가 / 언제 / 어떤 메시지 전송 / 수신을 하였는지 항상 기록되어야 한다.

(단, 이 때 Network에서의 오류는 일어나지 않고 상대방은 항상 온라인 상태라고 가정한다.)

1) 기능-모듈간 관계 리스트

이 때 조작자가 가지고 있는 기능-모듈간 관계 리스트에는 기능과 모듈에 연관된 요구사항들이 기록되어 있다. 다음은 해당 리스트의 일부분이다.

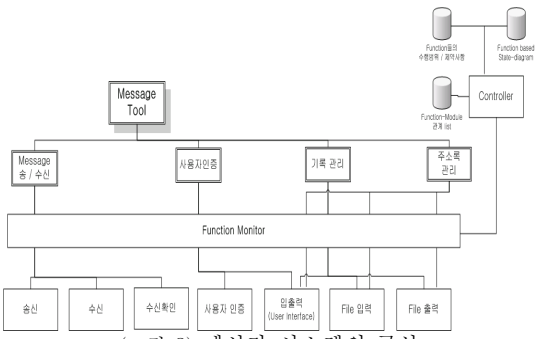
1. File 입력 - 기록 관리 : 요구사항 5

2 수신확인 - Message 송수신 : 요구사항 1

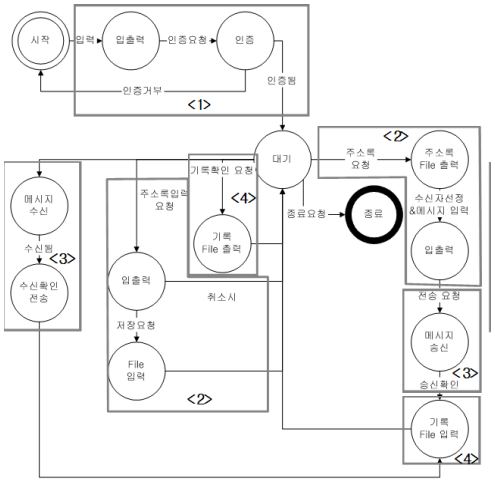
특정 모듈에서 특정 기능을 사용하였을 때 오류가 일어나면 해당 리스트를 적용시켜 어떤 요구사항이 영향을 받는지 파악 가능하다. 또, 3번과 같이 네트워크상에서 자신이 파악 가능하나 상대방에 알릴 필요가 있는 경우 이를 알림으로 상호 보완적인 역할이 가능하다.

2) 기능 기반 상태 다이어그램

조작자는 그림 4와 같은 기능 기반 상태 다이어그램을



(그림 3) 메신저 시스템의 구성



(그림 4) 기능 기반 상태 다이어그램

가지고 있으며 기능 감시자로부터 들어오는 보고를 통해 해당 상태가 정당한 상태인지 확인을 하고, 상태의 변환을 발생시킨다. 이 때, 그림 4의 <번호>는 각각의 모듈을 의미한다. 이를 State Diagram을 통해 확인 시 상태 1에서 상태 2로 변환되는 것을 어떤 모듈에서 어떤 기능을 사용했는지를 통해 파악이 가능하다. 또한 정당한 모듈이 적합한 기능을 사용하였는지에 관한 정당성도 파악 가능하고, 시스템의 전체적인 궤도를 또한 파악할 수 있다.

3) 기능들의 수행범위 / 제약사항

각 기능의 수행범위 / 제약사항은 특정 파일에 기록되어 있으며 그림 5는 해당 파일 중 수신 기능에 관련된 부분이다. 이는 컴파일 시 컨트롤러에 추가되어 기능 감시자에 의해 실행된다. 다음은 각 란에 대한 설명이다.

1. Module과 Func 란을 통해 어떤 모듈이 어떤 기능을 수행하는 지 확인한다.

2. Ptime을 통해 해당 기능의 수행 전 / 후에 검사하는지를 파악한다.

3. Type을 통해 어떤 값을 검사하는 지 결정한다.

4. Dtype을 통해 데이터 타입이 어떤 것인지 제한한다.

5. Size 1, 2를 통해 해당 값의 크기를 제한한다.

6. req 란에 추가적인 제약사항이 기록되어 있다.

예를 들면, 그림 5의 내용을 보고 입력 값이 검사된 후 해당 기능을 실행하기 직전 ti라는 현재 시간을 저장시켜 놓고 결과 값이 나오면 해당 기능의 수행시간을 검사해 요구사항 1의 수행 여부 역시 확인한다.

해당 시스템을 직접 실행해 본 결과는 다음 그림 6에 있다. 그림 6은 Send 명령 시 잘못된 매개변수 값을 줄

```
Module  Func   Ptime   Type    Dtype   size1   size2   req
M_S/R   Send   Before  input1  string  >0      <20
M_S/R   Send   Before  input2  string  >=0     <512    ti=System.currentTimeMillis();
M_S/R   Send   Before  ti      long    !=0
M_S/R   Send   After   output  bool                    cu=(System.currentTimeMillis()-ti)/1000;
M_S/R   Send   After   cu      long                    if(cu>3) rt_error();
```

(그림 5) 수신 기능의 수행범위 / 제약사항

```
Starting Message Manager
ID : well12
PW : ********
$Send
1. Angel
2. Devel
3. Fallen angel
Enter the receiver's number : 0
Variable input1 is not valid
Fail to make a transition from State 3
Related Module and Function are module Sender and function Send
Realated Requirements :
Requirement 1 : Couldn't send the message in 3 seconds
Requirement 4 : Send message / managing history failed
System failure occured
```

(그림 6) 오류 발생 시 실행 예제

경우 발생하는 문제이다. 당 시스템은 어떤 문제가 하나라도 발견될 경우 자동종료 되도록 설정되어있다. 그림 6과 같이 이상이 발생한 모듈과 기능을 파악 가능하고, 영향을 받은 요구사항 역시 파악이 가능하다.

위와 같은 Send 명령을 수행하기 위해 본 시스템의 적용과 미적용 사례 비교하면 표 1과 같다(실행시간 측정을 위해 10회의 평균을 표시하였다.)

<표 1> 모니터링 적용과 미적용의 비교

구 분	적 용	미적용
실행시간(평균)	47ms	45ms
요구사항 위반여부	파악 가능	파악 불가능
호출 정당성 여부	파악 가능	파악 불가능
이상 발생지역	파악 가능	파악 불가능

5. 결론 / 향후 연구과제

본 논문에서는 자가치유에서의 모니터링 기법에서 기능 단위의 이상부터 전체적인 시스템의 이상까지 발견가능하고 위반된 요구사항의 리스트까지 확인 가능한 시스템의 구성을 제안하였다. 또, 제안된 시스템의 구성과 이를 위한 평가를 통해 실질적으로 각 구성요소들이 어떠한 역할을 하는지 살펴보았다. 이를 통해 다음과 같은 결과를 얻을 수 있다.

1) 각 기능들의 수행범위 / 제약사항을 통하여 해당 기능을 상황에 따른 제약이 가능하고 특정 요구사항의 세부적인 감시가 가능하다.

2) 기능 기반 상태 다이어그램을 통하여 전체 시스템의 흐름을 감시하고 기능 감시자가 보고한 호출한 모듈과 호출된 기능의 정당성을 파악하여 부적절한 사용을 파악 가능하다.

3) 모듈-기능 관계 리스트를 통하여 어떤 이상이 발생하였을 경우 해당 이상이 영향을 미치는 요구사항들이 어떠한 것인지 간단하게 파악 가능하다.

4) 어떤 이상이 발생하였을 경우 기능 기반 상태 다이어그램과 기능 감시자가 보고한 내용에 따라 어떤 기능에서 문제가 일어났는지 파악 가능하다.

5) 조작자의 내용을 수정함으로 기능의 사용성을 확장하는 등의 시스템의 변경 / 관리의 용이성을 제공한다.

이를 통해 각 기능에서 일어나는 사항부터 시스템의 전체적인 흐름에 관련된 이상을 파악 가능하며 또한 해당 이상과 관련된 요구사항들을 간단히 파악 가능하다는 것을 알 수 있다. 그렇기에 자가치유의 모니터링 기법에 적용하여 진단 및 복구의 기능에 필요한 사전정보의 제공이 가능하다.

그렇지만 보다 정확하고 효율적인 시스템을 위한 향후 과제가 필요하다. 첫 번째로 시스템 디자인 시 상태 다이어그램 및 모듈-기능 관계를 제시해 주어야 하는 부하가 있다. 따라서 해당 부하를 줄일 수 있는 자동화 방법으로 시스템의 효율성을 높일 수 있다. 두 번째로 관계 리스트를 통해 파악하는 요구사항의 위반은 리스트에서 여러 개의 요구사항이 주어질 수 있다. 따라서 보다 정확한 위반사항은 사용자 혹은 관리자가 판단해야 하기에 이에 대한 자동화 방법으로 시스템의 정확성을 높일 수 있다.

참 고 문 헌

[1] A. G. Ganek, T. A. Corbi, "The dawning of the autonomic computing era", IBM System Journal pp. 5-18, 2003

[2] D. Ghosh , R. Sharman , H. R. Rao , S. Upadhyaya, "Self-healing systems - survey and synthesis", Elsevier, Decision Support System 42, pp. 2164-2185, 2007

[3] D. Garlan, S. W. Cheng, An-Cheng Huang, Bradley Schmerl, Peter Steenkiste, "Rainbow Architecture-Based Self-Adaptation with Reusable Infrastructure", IEEE Computer Society, pp. 46-54, 2004

[4] M. E. Shin, "Self-healing components in robust software architecture for concurrent and distributed systems", Elsevier, Science of Computer Programming 57, pp. 27-44, 2005

[5] M. Richters, M. Gogolla, "Aspect-Oriented Monitoring of UML and OCL Constraints", in Proc. of The 4th AOSD Modeling With UML Workshop, 2003

[6] W. N. Robinson, "A Requirements Monitoring Framework for Enterprise Systems", Requirements Engineering, Springer, Vol. 11 No. 1, pp. 17-41, 2006

[7] D. Tosi, "Research Perspectives in Self-healing systems, Department of IT", Systems and Communications, pp. 1-25, 2004

[8] D. S. Wile, A. Egyed, "An Externalized Infrastructure for Self-Healing System", IEEE, Computer Society, pp. 1-4, 2004

[9] M. Jiang, J. Zhang, D. Raymer, J. Strassner, "A Modeling Framework for Self-healing Software Systems", pp. 1-9,

[10] R. Westdijk, L. Rothkrantz, A.V. van Leijen, "Applying Requirements Monitoring for Autonomic Computing in a Combat Management System", in Proc. of Autotestcon 2007, pp. 349-358, 2007