자가 적응 모듈의 성능 개선과 오류 탐지를 위한 코드 자동 생성 기법

(An Automated Code Generation for Both Improving Performance and Detecting Error in Self-Adaptive Modules)

이 준 훈 † 박 정 민 †† 이 은 석 †††
(Joonhoon Lee) (Jeongmin Park) (Eunseok Lee)

요 약 오늘날 복잡해져 가는 컴퓨팅 환경에서 시스템에 발생하는 다양한 문제를 시스템 관리자가 직접 처리하는 것은 한계가 있다. 이 한계를 해결하기 위해서 시스템 스스로 상황을 인식하여 적절한 대응하는 능력을 갖는 것이 중요한 이슈가 되고 있다. 그러나 자가 적응 시스템을 생성하기 위해서는 많은 경험과 지식이 필요하다. 따라서 자가 적응 시스템 구축의 어려움이 문제가 되고 있다. 본 논문에서는 그러한 자가 적응 시스템의 구축을 용이하게 하기 위하여 자가 적응 시스템의 코드를 자동 생성하는 기법을 제안한다. 본 자가 적응 시스템은 기존 관련 연구에서 문제가 된 시스템의 리소스 과다 사용을 통한 비효율성과 바이러스와 같은 외부 요인에 의한 부정확한 동작에 대한 문제를 부분적으로 해결한다. 본 논문에서는 평가를 위하여 비디오 회의 시스템에서 사용하는 파일 전송 모듈에 제안 방법론을 적용하였다. 개발자가 추가로 작성한 코드의 길이, 개발자가 만든 클래스의 수, 개발 시간을 제안 방법론 적용 전과 후를 비교하여 그 유효성을 확인하였다.

키워드 : 자가 적응, 자가 치유, 목표 그래프, 동작 스위치, 오류 탐지

Abstract It has limits that system administrator deals with many problems occurred in systems because computing environments are increasingly complex. It is issued that systems have an ability to recognize system's situations and adapt them by itself in order to resolve these limits. But it requires much experiences and knowledge to build the Self-Adaptive System. The difficulty that builds the Self-Adaptive System has been problems. This paper proposes a technique that generates automatically the codes of the Self-Adaptive System in order to make the system to be built more easily. This Self-Adaptive System resolves partially the problems about ineffectiveness of the exceeded usage of the system resource that was previous research's problem and incorrect operation that is occurred by external factors such as virus. In this paper, we applied the proposed approach to the file transfer module that is in the video conferencing system in order to evaluate it. We compared the length of the codes, the number of Classes that are created by the developers, and development time. We have confirmed this approach to have the effectiveness.

Key words : Self-Adaptive, Self-Healing, Goal graph, Activation switch, Error detection

· 본 연구는 지식경제부의 유비쿼터스 컴퓨팅 및 네트워크 원천 기반 기술개발사업(08B3-B1-10M), 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업 IITA-2008-(C1090-0801-0046), 교육과학기술부의 특정기초연구사업 R01-2006-000-10954-0의 연구 결과로 수행되었음
· 이 논문은 2008 한국 소프트웨어공학 학술대회에서 '자가 적응 모듈의 성능 개선과 오류 탐지를 위한 코드 자동 생성 기법'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 성균관대학교 전자전기컴퓨터공학과
trsprs@ece.skku.ac.kr
†† 학생회원 : 성균관대학교 컴퓨터공학과
jmpark@ece.skku.ac.kr
††† 종신회원 : 성균관대학교 정보통신공학부 교수
eslee@ece.skku.ac.kr

논문접수 : 2008년 4월 17일
심사완료 : 2008년 8월 27일

정보과학회논문지: 소프트웨어 및 응용 제35권 제9호(2008.9)

1. 서 론

컴퓨팅 환경이 점점 복잡해지면서 시스템에서 발생할 수 있는 문제들을 분석하고 그것들을 해결하는 것이 점점 어려워지고 있다[1]. 이러한 문제들을 해결하기 위한 첫 번째 방법은 인간이 직접 시스템을 관리하는 것이고, 두 번째 방법은 시스템 스스로가 문제를 인식하여 해결하게 하는 방법이다.

인간이 직접 시스템을 관리하는 방식에서는 다양한 컴퓨터 시스템을 효율적으로 관리할 수 있는 전문가의 필요성이 증가하고 있지만, 이용 가능한 인적자원과 효과적인 비용관리의 측면에서 볼 때 인간에 의한 시스템의 관리는 명확한 제약이 있다[2]. 심지어는 전체 컴퓨터 시스템의 오류 중 약 40%가 관리자의 오류에 의한 것이라고 하니[3], 전문 관리자에 의존하는 현재의 시스템 관리 방식은 개선되어야 할 필요가 있다. 인간 중심의 시스템 관리 방식을 개선하기 위한 방법으로 자가 치유에 관한 연구가 중요한 이슈가 되고 있다.

자가 치유 연구는 시스템의 문제를 스스로 인식, 진단하여 문제에 대한 전략을 적용하여 시스템이 자율적으로 복구하도록 하는 것이다[4,5]. 오늘날 중대한 시스템(critical system)들은 목표 시스템의 신뢰성, 강건성, 가용성 확보를 위해 추가적인 자가 적응 모듈의 설계와 비용이 고려되고 있다. 이러한 측면에서 중대한 시스템 내부에 존재하는 자가 적응 모듈로 인해서 발생되는 성능 저하 문제가 해결되어야 한다.

본 연구 그룹에서는 자가 적응 모듈에서 소비되는 추가적인 리소스를 줄이기 위한 방법으로 동작 스위치를 제안하였다. 이 스위치는 사용되지 않는 불필요한 자가 적응 컴포넌트를 판별하기 위한 수단이다. 그러나 동작 스위치를 적용하는 과정에서 다음과 같은 문제점이 있었다.

- 비정상적인 외부 요인에 의해, 자가 적응 컴포넌트의 상태가 동작 스위치와 일치하지 않아 자가 적응 모듈이 정상적인 동작을 하지 않는다. 예를 들어, 메모리의 특정 부분을 변경시킬 수 없게 하는 바이러스가 자가 적응 모듈에 영향을 주는 경우, 동작 스위치와 컴포넌트의 상태가 일치하지 않을 수 있다. 따라서 컴포넌트가 동작해야 하는 상황임에도 동작하지 않을 수 있다.
- 동작 스위치를 이용한 자가 적응 모듈을 생성하기 위해 개발자의 추가적인 노력이 필요하다. 예를 들어, 이미 만들어진 자가 적응 모듈에 동작 스위치를 적용하기 위해서는 동작 스위치에 대한 사항을 고려하여 전체 구조를 다시 설계해야 한다.

본 연구에서는 이러한 문제를 해결하기 위해서 이전 연구의 확장을 통해 목표 그래프를 사용하여, 자가 적응 모듈의 오류를 탐지하고, 성능 향상을 위해 동작 스위치를 사용하는 개선된 자가 적응 모듈을 자동 생성한다.

본 논문에서는 평가를 위하여 제안 방법론을 화상 회의 시스템의 파일 전송 모듈에 적용하였다. 파일 전송을 위한 모듈의 목표와 동작 중에 의심스러운 행동을 탐지하여 적응하려는 자가 적응 모듈의 목표를 입력하여 자동으로 자가 적응 모듈을 생성한다. 이 자가 적응 모듈은 동작 스위치를 사용하여 리소스 사용을 줄일 수 있으며, 외부 요인에 의한 컴포넌트의 오작동을 막을 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 관해 기술하고, 3장에서는 본 논문에서 제안하는 방법론을 소개한다. 4장에서는 본 논문에서 제안하는 내용을 실제로 적용하기 위한 도구를 구현하고 평가 한다. 마지막으로 5장에서 결론을 기술한다.

2. 관련 연구

본 장에서는 Wile의 자가 적응 아키텍처[6], 우리의 이전 연구인 동작 스위치를 이용한 자가 적응 모듈, 외부 요인에 의한 자가 적응의 오류를 탐지하는 구조[7]와 Stelling의 하트비트 모니터링(Heartbeat monitoring)[8], 목표 그래프를 이용한 개발 방법론[9]에 대해 살펴보고, 그에 대한 장점과 단점을 기술한다.

2.1 자가 적응 아키텍처

Wile[6]은 프루브 계층(probe layer), 게이지 계층(gauge layer), 제어 계층(control layer), 그리고 실행 계층(execution layer)과 같은 네 가지 계층을 사용하여 시스템의 의심스러운 행동을 탐지, 치유한다. *프루브 계층*에서는 대상 시스템을 감시하고, 감시를 통해 수집된 정보는 프루브 버스(probe bus)를 통해 *게이지* 계층에 전달된다. 이러한 정보는 *게이지 계층*에서 분석되어 현재의 상태를 파악하는데 사용된다. 분석된 상태 정보는 게이지 버스(gauge bus)를 통해 제어 계층에 전달된다. *제어 계층*은 분석된 정보를 통해 어떠한 전략을 적용해야 하는지를 결정하게 된다[10]. 전략에 대한 결정은 시스템 스스로 이루어지기도 하나, 필요에 따라 사용자에 의해 결정되도록 하기도 한다. 결정이 내려지면 *실행 계층*에 전달되어 대상 시스템에 적용하여 시스템이 현재의 상태에 적응할 수 있도록 한다.

그러나 이러한 과정과 관련된 많은 컴포넌트를 필요로 하며 추가적인 리소스가 사용되기 때문에 현재 상태에서 필요하지 않은 리소스의 낭비를 야기한다.

2.2 동작 스위치를 이용한 성능 개선

중대한 시스템(critical system)들은 목표 시스템의 신뢰성, 강건성, 가용성 확보를 위해 추가적인 자가 적

응 모듈의 설계와 비용이 고려된다. 이러한 측면에서 중대한 시스템 내부에 존재하는 자가 적응 모듈이 야기하는 성능 저하 문제는 해결되어야 한다.

우리의 이전 연구에서는 자가 적응 모듈[7]의 단계별 동작(주소 허가 판단, 예상 시간 측정, 파일 전송, 주소 차단 및 파일 삭제)을 위해 동작 스위치를 사용하였다. 동작 스위치는 이렇게 나누어진 각 단계들을 언제 작동해야 하는지를 판별하는 기준이다. 각 단계에서, 필요한 컴포넌트는 작동을 시작하고 그렇지 않은 컴포넌트는 작동을 멈춘다. 이를 통해, 필요하지 않은 컴포넌트가 리소스를 사용하는 것을 줄일 수 있다. 자가 적응 모듈은 추가적인 리소스를 사용하기 때문에 리소스의 사용을 얼마나 줄이느냐에 따라 성능이 좌우된다. 동작 스위치를 통해 리소스 사용을 줄일 수 있기 때문에 결과적으로 시스템의 성능을 향상시킬 수 있다.

하지만 이전 연구에서는 특정 시스템에 국한하여 기술하였기 때문에 추상화된 기법을 제공하지 않는다. 또한 개발자가 동작 스위치를 생성하기 위해서는 모든 코드를 직접 작성해야 하는 추가적인 노력이 필요하게 된다. 따라서 우리의 확장 연구는 목표 그래프를 기반한 추상화 기법을 제공하고, 이를 바탕으로 오류 탐지를 위한 코드를 자동으로 생성한다.

2.3 오류 탐지 모니터링 방법론

Stelling[8]은 컴포넌트의 주기적인 신호를 통해 컴포넌트의 상태가 정상인지 비정상인지를 판단하는 방법을 제안하였다. 컴포넌트가 *"I am alive."*라는 신호를 주기적으로 보내면 컴포넌트를 감시하고 있던 모니터는 신호를 통해 컴포넌트가 정상적으로 동작하고 있음을 알게 된다. 만약 모니터가 신호를 감지하지 못한다면 해당 컴포넌트는 비정상적인 동작을 하고 있음을 의미한다.

이 방법론은 시스템이나 컴포넌트를 감시하는 모니터가 직접 상태를 확인하지 않기 때문에 자가 적응 모듈의 리소스 소모량을 줄일 수 있는 장점이 있다.

2.4 동작 스위치의 오류 탐지

본 연구 그룹에 의해 이전에 개발된 동작 스위치 오류 탐지[7]는 자가 적응 모듈에서 사용되는 동작 스위치의 스위칭 정확도를 모니터링 하기 위한 구조이다. 이 구조는 그림 3의 스위치 템플릿 생성자(Switch Template Generator, 굵은 사각형으로 표시)가 템플릿 코드를 생성하기 위한 입력 중 하나로 사용된다. 추가적인 구조를 사용하여 자가 적응 모듈을 좀 더 안전한 상태로 실행되게 한다. 이러한 컴포넌트들은 다음과 같은 목적을 가진다.

- 자가 적응 모듈에서 발생하는 비정상적인 상태를 감지한다.
- 비정상적인 상태를 치유하기 위한 전략을 수립하여 적용한다.
- 자가 치유 중에도 컴포넌트의 작업을 계속 하도록 대체 컴포넌트를 제공한다.

2.5 목표 그래프를 이용한 코드 생성 기법

목표 그래프를 이용한 코드 생성 기법 [9]에서는 개발자가 자가 적응 모듈을 직접 작성하고 관리하는 노력을 덜기 위하여 목표 그래프를 이용한 외부 요인 평가 기법 [11]의 목표 그래프를 응용하였다. 이것은 자가 적응 모듈의 성능을 개선하는 방법을 추상화한다.

그림 1은 그림 2와 같은 목표 그래프를 개발자가 그린 후, 자가 적응 모듈을 생성하는 과정을 나타내고 있다. 이러한 과정을 통해 생성된 코드는 개발자의 수정을 거쳐 최종적으로 자가 적응 모듈을 완성한다.

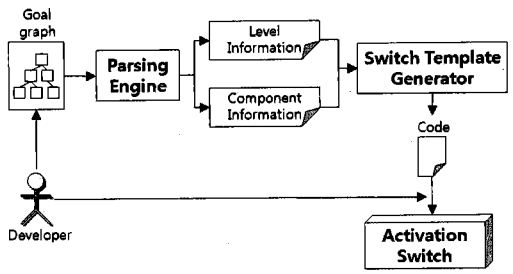


그림 1 목표 그래프를 이용한 코드 생성 구조

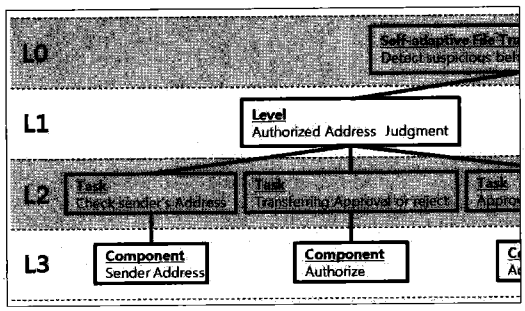


그림 2 목표 그래프의 예

3. 제안 사항

본 논문에서는 기존의 연구들을 통해 분석된 장점과 단점을 부분적으로 이용, 해결하기 위해, 목표 그래프를 확장하여 자가 적응 모듈의 성능을 개선하는 추상적 개선 기법, 에러 탐지를 위한 모듈[7]을 자동 생성하는 자동 생성 기법을 제안한다. 본 논문에서는 에러 탐지 구조를 추가한 자동 코드 생성에 초점을 두므로 에러 탐지를 위한 모듈에서 제안한 재사용 프로세스는 사용하지 않는다.

그림 3은 개발자가 그린 목표 그래프를 이용하여 자가 적응 모듈을 생성하는 과정을 나타내고 있다. 이러한

표 1 제안 시스템 구조의 기능 요약

이름	기능
Goal graph designer (3.1절)	목표 그래프를 그릴 수 있는 도구
Parsing Engine (3.2절)	목표 그래프를 통해 필요한 정보들을 수집
Activation Information	동작 스위치를 위한 정보
Component Information	모듈 내의 컴포넌트 정보
Layer Information	컴포넌트가 속한 계층에 대한 정보
Switch Template Generator (3.4절)	입력된 정보로 템플릿 코드를 생성

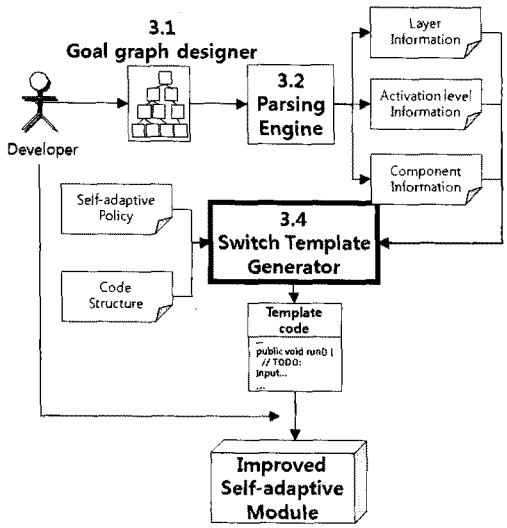


그림 3 제안 시스템 구조

과정을 통해 생성된 코드는 개발자의 수정을 거쳐 최종적으로 자가 적응 모듈을 완성하게 된다.

표 1은 그림 3의 각 컴포넌트의 기능에 대해 설명한다. 이 구조는 두 가지의 목적을 가지고 있다. 먼저, 개발자가 그린 목표 그래프를 통해 자가 적응 모듈의 스위치를 생성하기 위한 정보들을 입력 받아 동작 스위치를 생성한다. 두 번째로 생성된 동작 스위치와 코드의 구조, 자가 적응 정책 등을 적용하여 자가 적응 모듈의 템플릿 코드를 생성한다.

3.1 목표 그래프 설계

표 1에서 목표 그래프 디자이너(Goal graph designer)의 역할은 개발자가 자가 적응 모듈의 단계를 잘 표현하도록 목표 그래프를 그릴 수 있게 하는 것이다. 표 2는 목표 그래프의 구조를 나타낸다. 목표 그래프는 최상위에 목표(Goal) 노드를 가진다. 목표 노드는 자식 노드이며 단계(Level) 노드를 가져야 한다. 이러한 구조를 통해 해당 목표를 위한 단계가 어떤 것이 있는지를 판단할 수 있다. 또한 단계 노드는 작업(Task) 노드를 자식 노드로 가진다. 작업 노드는 현재 단계에서 수행하게 될 작업들을 나타낸다.

따라서 단계 노드는 작업 노드를 여러 개 가질 수 있게 되고, 작업 노드는 작업을 달성하는데 필요한 컴포넌트(Component) 노드를 자식 노드로 가지게 된다. 이 때, 해당 컴포넌트가 자가 적응 계층(프루브 계층, 게이지 계층, 제어 계층, 실행 계층)중 어디에 속하는지를 표시하기 위한 계층(Layer) 노드를 추가하였다. 표 2의 L3은 자가 적응 아키텍처의 계층을 나타내며, 작업 노드와 컴포넌트 노드 사이에 위치한다. 이러한 구조를 통해 하나의 목표를 달성하기 위해서 필요한 단계, 작업, 자가 적응 계층, 컴포넌트들을 구조적으로 나타낼 수 있다.

그림 4는 표 2에서 제시한 *목표 그래프의 구조를 시각화하여 보여주기 위한 예*이며 목표 그래프를 이용한 자가 적응 모듈 생성 방법 [9]에서 제안한 파일 전송 자가 적응 모듈의 단계를 제안한 목표 그래프를 통해 그린 결과이다. 최상위에 위치한 목표를 통해 파일 전송 중에 의심스러운 행동을 탐지하는 것이 이 모듈의 역할임을 나타낼 수 있다.

또한 L1에서 해당 목표를 위해 허가된 주소인지를 판단하는 단계를 나타낸다. 허가된 주소인지를 판단하기 위해서는 여러 작업들이 필요한데, L2에서 그러한 작업들을 나열하고 있다. L3에서는 작업을 수행하기 위해 사용되는 자가 적응 계층을 표시하며 해당 계층에서 사

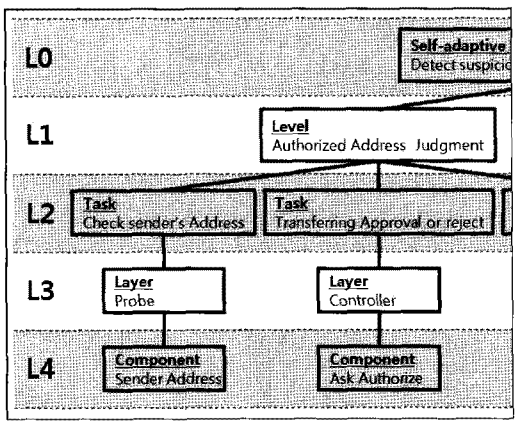


그림 4 확장된 목표 그래프의 예

표 2 목표 그래프의 구조

계층	이름	속성
L0	Goal	자가 적응 모듈의 목표
L1	Level	자가 적응 모듈의 단계
L2	Task	자가 적응 모듈에서 각 단계를 위한 작업
L3	Layer	자가 적응 아키텍처의 계층
L4	Component	자가 적응 모듈에서 각 작업을 위한 컴포넌트

표 3 목표 그래프에서 노드들의 속성

종류	속성	설명
Goal	이름	자가 적응 모듈의 이름
	내용	자가 적응 모듈의 목표
Level	이름	고정된 텍스트: Level
	내용	단계의 목표
Task	이름	고정된 텍스트: Task
	내용	작업의 내용
Layer	이름	고정된 텍스트: Layer
	내용	자가 적응 아키텍처의 계층
Component	이름	고정된 텍스트: Component
	내용	컴포넌트 이름

용되는 컴포넌트들을 L4에서 표시하고 있다. 예에서 볼 수 있듯이, 각 노드는 여러 개의 자식 노드를 가질 수 있다. 그림 4의 예에서, L3와 L4는 노드가 하나씩 연결되어 있지만, 작업에 따라 여러 개의 계층과 컴포넌트가 연결될 수 있다.

표 3은 목표 그래프에서 각 노드에 입력되어야 할 속성들을 나타낸다. 각 노드는 공통적으로 이름과 내용과 관련된 속성을 가진다. 주목할 것은 계층 노드를 통해 컴포넌트의 계층 종류와 관련된 속성을 추가로 가진다는 것이다. 자가 적응 모듈에서 사용하는 네 가지 계층[6]에서 해당 컴포넌트가 어느 계층에 위치하는지를 나타낸다. 이를 통해 해당 컴포넌트가 어떤 버스와 연결되는지를 표현할 수 있다.

3.2 파싱 엔진(Parsing Engine)

본 논문은 개발자나 관리자가 표 2의 구조에 목표 그래프를 올바르게 그렸다고 가정한다. 파싱 엔진은 목표 그래프를 분석하여 자가 적응 모듈을 구성하는데 필요한 정보들을 추출한다. 표 3에 나타나 있는 속성들은 파싱 엔진에 의해 추출되어 다음 단계로 전달된다. 정상적으로 그려진 목표 그래프를 통해 추출된 정보는 다른 컴포넌트들이 사용할 수 있는 형태로 저장된다.

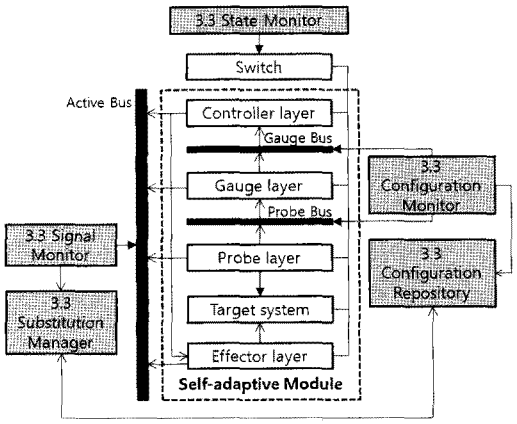


그림 5 자가 적응 모듈의 오류 탐지를 위한 구조

3.3 오류 탐지를 위한 구조의 개선

그림 5는 자가 적응 모듈의 오류 탐지를 기반으로 목표 그래프를 이용하여 소프트웨어 구조가 자동 생성할 수 있도록 수정한 것이다. 점선은 자가 적응 모듈의 아키텍처이다. 이러한 아키텍처에서 오류 탐지를 위한 컴포넌트를 추가하였다. 이 컴포넌트들은 다음의 목적을 가진다.

- 자가 적응 모듈에서 발생하는 비정상적인 상태를 감지한다.
- 비정상적인 상태를 치유하기 위한 전략을 적용한다.
- 자가 치유 중에도 컴포넌트의 작업을 계속 하도록 대체 컴포넌트를 제공한다.

표 4에서는 각 컴포넌트들의 간략한 기능을 요약하고 있다. 각 컴포넌트들의 세부적인 사항은 다음과 같다.

- 상태 모니터(State Monitor): 이 컴포넌트는 동작 스위치의 상태를 모니터링 한다. 모니터링을 하기 위해서는 스위치에 대한 정보가 있어야 하는데 이것은 목표 그래프를 통해서 얻어진 정보를 사용한다(표 1의 레벨 정보(Level Information), 동작 정보(Activation Information), 계층 정보(Layer Information)).

표 4 오류 탐지를 위한 컴포넌트 기능 요약

이름	기능	필요한 정보
State Manager	1. 현재 파일 전송 2. 컴포넌트 동작 스위치를 모니터링	1. 자가 적응 컴포넌트들의 계층 정보 2. 동작 스위치의 상태
Configuration Monitor (CM)	1. Probe Bus, Gauge Bus를 모니터링 2. Bus로 전달되는 값을 CR에 저장	1. Bus에 관한 정보
Configuration Repository	1. CM에서 모니터링 한 값을 저장	1. 모니터링 한 값 2. 모니터링 한 대상
Signal Monitor	1. 자가 적응 모듈의 컴포넌트가 동작을 시작하는 신호를 모니터링	1. 동작 스위치의 상태
Substitution Manager	1. 모듈의 단계별로 대체 컴포넌트를 생성하고 관리 2. 자가 치유 전략을 적용 시 대체 컴포넌트 제공	1. 컴포넌트 정보 2. 대체할 컴포넌트 종류
Component Pool (CP)	1. Substitution Manager에 의해 생성된 대체 컴포넌트를 보관, 관리	1. 컴포넌트 정보

- 신호 모니터(Signal Monitor): 동작 스위치를 이용한 자가 적응 컴포넌트는 동작 스위치가 켜질 때마다 동작을 시작하고, 꺼질 때 동작을 멈춘다[9]. 만약 동작 스위치가 켜졌을 때, 컴포넌트가 동작하지 않으면 오류 상황으로 간주하게 된다. 하지만 컴포넌트가 실제로 동작하는지를 체크하는 것이 없다. 따라서 이 방법은 컴포넌트가 동작을 시작할 때마다 활성화 버스(Active Bus)로 동작을 시작했음을 알리게 된다. 시그널 모니터는 동작 스위치가 켜졌을 때 이러한 신호가 있는지를 모니터링 하고, 일정 시간 내에 신호가 없을 시 해당 컴포넌트는 오류가 발생했음을 탐지한다.
- 대체 매니저(Substitution Manager): 오류 탐지를 위한 방법론에서는 자가 치유를 위한 전략으로 컴포넌트를 대체하는 것을 사용한다. 본 논문의 제안 사항에서도 이러한 치유 전략을 사용한다.
- 구성 모니터(Configuration Monitor): 대체 컴포넌트를 이용한 자가 치유 전략을 사용하기 위해서는 대체 컴포넌트에서 사용할 정보를 알고 있어야 한다. 구성 모니터는 자가 적응 모듈에 존재하는 버스들을 모니터링 하여 다음 계층으로 넘어가는 정보들을 저장한다. 저장하는 위치는 구성 저장소(Configuration Repository)이다.

3.4 스위치 템플릿 생성자

스위치 템플릿 생성자(Switch Template Generator)는 목표 그래프를 통해 추출된 정보와 코드 구조, 자가 적응 정책 등을 입력 받아, 개선된 자가 적응 모듈의 템플릿 코드를 생성한다[9]. 코드 구조는 오류 탐지를 위한 구조를 바탕으로 작성된 것이다. 본 논문에서는 자가 적응 정책 중, 대체 컴포넌트를 이용한 방법[1]을 사용한다. 템플릿 코드는 성능을 개선하기 위한 스위치를 포함하며, 자가 적응 모듈 자체에 오류가 있는지를 탐지하기 위한 컴포넌트들도 포함된다. 수정된 목표 그래프에서는 각 컴포넌트들이 속하는 계층 정보를 추가로 입력하고, 이 정보들은 오류 탐지를 위한 코드를 생성할 때 사용된다.

3.5 자가 적응 모듈 코드 생성 프로세스

그림 6은 자가 적응 모듈의 코드를 생성하기 위한 과정을 보여 준다. 1) 목표 그래프를 그리고, 2) 목표 그래프에서 정보를 추출한다. 3) 개발자는 자가 적응 모듈의 오류 탐지를 위해 필요한 정보를 작성하고, 4) 추출된 정보들을 이용하여 템플릿 코드를 생성한다. 5) 생성된 템플릿 코드는 개발자에 의해 수정된 후, 자가 적응 모듈이 완성된다.

- 목표 그래프를 작성(Step 1): 목표 그래프를 그린다. 본 논문에서는 개발자에 의해 그려진 목표 그래프가 정확하게 그려졌다고 가정한다. 따라서 별도의 그래프 체크 작업은 하지 않는다.
- 그래프의 정보 추출(Step 2): 목표 그래프에서 컴포넌트들의 종류, 자가 적응 계층에 어떤 컴포넌트들이 속하는지, 목표를 달성하기 위해서 어떠한 단계로 동작하는지 등에 관한 정보를 추출한다.
- 오류 탐지를 위한 정보 열기(Step 3): 본 논문에서는 외부 상황에 의한 자가 적응 모듈의 오류를 탐지하는 기법의 일부를 수정하여 사용한다. 따라서 오류 탐지를 위해서는 자가 적응 모듈 자체에 대한 부가적인 내용과 오류가 발생했을 때의 자가 치유 정책도 필요하다. 이 단계에서는 미리 작성된 정책을 이용한다. 본 논문에서는 대체 컴포넌트를 사용하는 자가 치유

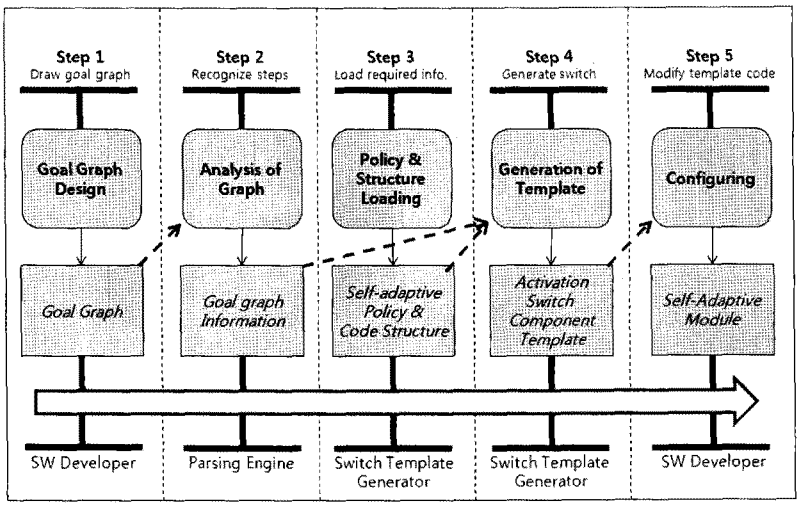


그림 6 코드 생성을 위한 프로세스

정책에만 한정하여 적용한다. 또한 이러한 치유 정책은 본 논문의 범위에 있지 않으므로 자세한 설명은 생략 한다.

- 템플릿 코드 생성(Step 4): Step 2와 Step 3에서 생성된 정보들은 이 단계에서 스위치 템플릿 생성자에 의해 사용된다. 미리 입력되어 있는 코드와 이러한 정보들을 혼합하여 자가 적응 모듈을 위한 템플릿 코드를 생성한다.
- 템플릿 코드 수정(Step 5): 개발자는 Step 4에서 생성된 템플릿 코드에서 TODO 메시지에 따라 자가 적응 모듈을 수정한다. 이러한 메시지가 있는 부분에 필요한 코드를 작성한다. 이러한 과정을 통해 자가 적응 모듈이 완성된다.

4. 구현 및 평가

본 논문에서는 목표 그래프를 이용한 개발 기법 [9]의 구현을 확장하였고, 구현을 위해서 Eclipse의 PDE (Plug-in Development Environment)를 이용하였다. 또한 목표 그래프를 설계하는 플러그-인을 구현하기 위하여 EMF(Eclipse Modeling Framework)와 GEF (Graphical Editing Framework)를 바탕으로 한 GMF (Graphic Modeling Framework)를 이용하였다. GMF를 통해 목표 그래프 설계를 할 수 있는 플러그-인을 구현하였고, 구현된 플러그-인을 통해 동작 스위치를 이용한 소개한 파일 전송 모듈의 목표 그래프를 설계하였다. 그림 7은 이러한 플러그-인을 이용한 설계 화면을 나타내고 있다.

설계된 목표 그래프는 XMI (XML Meta-Interface)의 형태로 저장된다. 파싱 엔진은 이 XMI 파일을 입력으로 하여 필요한 정보들을 미리 정의한 자료 구조에 저장한다. 스위치 템플릿 생성자는 이 자료 구조와 미리 작성된 자가 적응 정책(Self-Adaptive policy), 코드 구조(Code structure)를 입력으로 외부 상황에 의해 야기

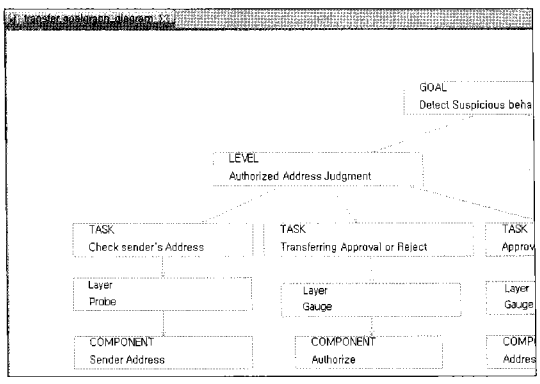


그림 7 목표 그래프를 위한 플러그인

```
SenderAddress.java
...
public void run() {
   while(on) {
   // TODO: Write down behavior of component
...
ProbeBus.java
...
 // TODO: Input your variables.
   // For example,
   // private String sendAddress;// Sender's address

   // TODO: If you write down any variables, you should create
   // methods for these variables. YOU MUST CREATE setter/getter method.
...
SwManager.java
...
   // Setting values
   // TODO:You can write some variables for Probe Bus and Gauge Bus.
...
 public void create() {

      cSenderAddress = new SenderAddress();
      cSenderAddress.setProbeBus(probeBus);
      cSenderAddress.setGaugeBus(gaugeBus);

      cAddressEstimation = new AddressEstimation();
      cAddressEstimation.setProbeBus(probeBus);
      cAddressEstimation.setGaugeBus(gaugeBus);
...
```

그림 8 생성된 템플릿 코드의 일부

된 모듈 내부의 오류 탐지를 가능하게 하였고, 동시에 동작 스위치를 통하여 성능을 개선한 자가 적응 모듈의 템플릿 코드의 생성이 가능하였다.

그림 8은 생성된 템플릿 코드의 일부를 보여 준다. 여기서 TODO 메시지를 확인할 수 있는데, 개발자는 각 컴포넌트의 기능을 위한 코드를 작성해야 한다. 개발자에게 의해 수정된 코드는 최종적으로 완성된 자가 적응 모듈이 된다. 이런 과정을 통해 완성된 자가 적응 모듈을 비디오 회의 시스템의 파일 전송 모듈에 추가하였다.

그림 9는 개발자가 템플릿 코드를 수정하여 자가적응 모듈을 완성하여 실행한 결과이다. 오류 탐지 확인을 위하여 컴포넌트가 동작하여야 하는 상황에서 동작하지 않는 환경을 가상으로 만들었다. 그림 9의 동그라미 부분은 이러한 오류를 탐지하여 적응 전략이 적용되는 것을 보여준다.

그림 10은 제안 사항을 적용하기 전과 후의 개발자의 노력을 비교한 결과를 보여준다. 적용 전을 100%로 보았을 때, 제안 사항을 적용하면 얼마나 노력이 감소하는지 알 수 있다[12]. 첫 번째로, LOC(Line Of Code)를 비교하여 미리 생성된 코드를 통해 개발자가 추가로 생성해야 할 코드의 양이 51%정도 줄어드는 것을 확인할

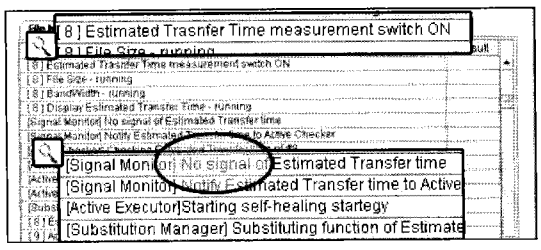


그림 9 오류 탐지를 통한 스위치 작동

- 신호 모니터(Signal Monitor): 동작 스위치를 이용한 자가 적응 컴포넌트는 동작 스위치가 켜질 때마다 동작을 시작하고, 꺼질 때 동작을 멈춘다[9]. 만약 동작 스위치가 켜졌을 때, 컴포넌트가 동작하지 않으면 오류 상황으로 간주하게 된다. 하지만 컴포넌트가 실제로 동작하는지를 체크하는 것이 없다. 따라서 이 방법은 컴포넌트가 동작을 시작할 때마다 활성화 버스(Active Bus)로 동작을 시작했음을 알리게 된다. 시그널 모니터는 동작 스위치가 켜졌을 때 이러한 신호가 있는지를 모니터링 하고, 일정 시간 내에 신호가 없을 시 해당 컴포넌트는 오류가 발생했음을 탐지한다.
- 대체 매니저(Substitution Manager): 오류 탐지를 위한 방법론에서는 자가 치유를 위한 전략으로 컴포넌트를 대체하는 것을 사용한다. 본 논문의 제안 사항에서도 이러한 치유 전략을 사용한다.
- 구성 모니터(Configuration Monitor): 대체 컴포넌트를 이용한 자가 치유 전략을 사용하기 위해서는 대체 컴포넌트에서 사용할 정보를 알고 있어야 한다. 구성 모니터는 자가 적응 모듈에 존재하는 버스들을 모니터링 하여 다음 계층으로 넘어가는 정보들을 저장한다. 저장하는 위치는 구성 저장소(Configuration Repository)이다.

3.4 스위치 템플릿 생성자

스위치 템플릿 생성자(Switch Template Generator)는 목표 그래프를 통해 추출된 정보와 코드 구조, 자가 적응 정책 등을 입력 받아, 개선된 자가 적응 모듈의 템플릿 코드를 생성한다[9]. 코드 구조는 오류 탐지를 위한 구조를 바탕으로 작성된 것이다. 본 논문에서는 자가 적응 정책 중, 대체 컴포넌트를 이용한 방법[1]을 사용한다. 템플릿 코드는 성능을 개선하기 위한 스위치를 포함하며, 자가 적응 모듈 자체에 오류가 있는지를 탐지하기 위한 컴포넌트들도 포함된다. 수정된 목표 그래프에서는 각 컴포넌트들이 속하는 계층 정보를 추가로 입력하고, 이 정보들은 오류 탐지를 위한 코드를 생성할 때 사용된다.

3.5 자가 적응 모듈 코드 생성 프로세스

그림 6은 자가 적응 모듈의 코드를 생성하기 위한 과정을 보여 준다. 1) 목표 그래프를 그리고, 2) 목표 그래프에서 정보를 추출한다. 3) 개발자는 자가 적응 모듈의 오류 탐지를 위해 필요한 정보를 작성하고, 4) 추출된 정보들을 이용하여 템플릿 코드를 생성한다. 5) 생성된 템플릿 코드는 개발자에 의해 수정된 후, 자가 적응 모듈이 완성된다.

- 목표 그래프를 작성(Step 1): 목표 그래프를 그린다. 본 논문에서는 개발자에 의해 그려진 목표 그래프가 정확하게 그려졌다고 가정한다. 따라서 별도의 그래프 체크 작업은 하지 않는다.
- 그래프의 정보 추출(Step 2): 목표 그래프에서 컴포넌트들의 종류, 자가 적응 계층에 어떤 컴포넌트들이 속하는지, 목표를 달성하기 위해서 어떠한 단계로 동작하는지 등에 관한 정보를 추출한다.
- 오류 탐지를 위한 정보 열기(Step 3): 본 논문에서는 외부 상황에 의한 자가 적응 모듈의 오류를 탐지하는 기법의 일부를 수정하여 사용한다. 따라서 오류 탐지를 위해서는 자가 적응 모듈 자체에 대한 부가적인 내용과 오류가 발생했을 때의 자가 치유 정책도 필요하다. 이 단계에서는 미리 작성된 정책을 이용한다. 본 논문에서는 대체 컴포넌트를 사용하는 자가 치유

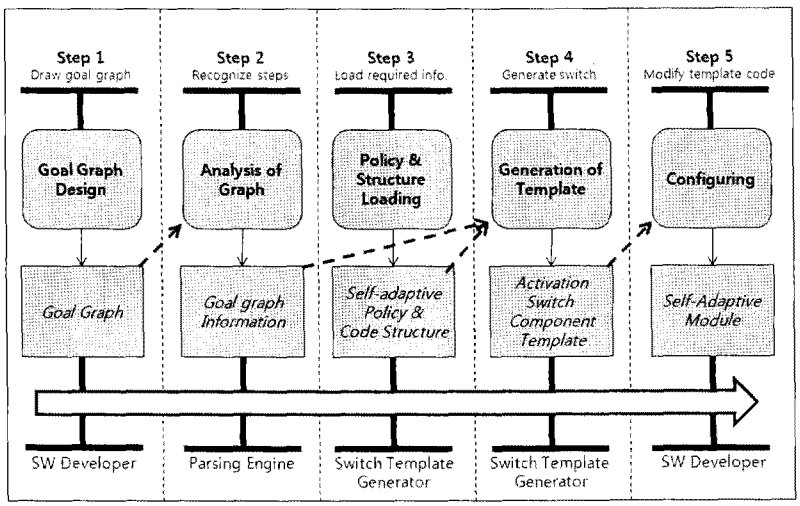
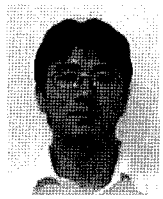


그림 6 코드 생성을 위한 프로세스

[6] David S. Wile, Alexander Egyed, "An Externalized Infrastructure for Self-Healing Systems," In Proceedings of the 4th Working IEEE/IFIP Conference on Software Architecture, pp.285-288, Sep., 2004.
[7] 이준훈, 이희원, 박정민, 정진수, 이은석, "자가 적응 모듈의 오류 탐지와 재사용," 제 34회 한국정보과학회 추계학술대회, pp. 247-252, Oct., 2007.
[8] P. Stelling, I. Foster, C. Kesselman, C. Lee, and G. V. Laszewski, "A fault detection service for wide area distributed computations," In Proceedings of 7th High Performance Distributed Computing, pp.268-278, Jul., 1998.
[9] Jeongmin Park, Joonhoon Lee, and Eunseok Lee, "Goal graph based Performance Improvement for Self-Adaptive Modules," ACM SIGKDD International Conference on Ubiquitous Information Management and Communication (ACM SIGKDD ICUIMC2008), pp.68-72, Feb., 2008.
[10] David Garlan, Bradley Schmerl, and Jichuan Chang, "Using Gauges for Architecture-Based Monitoring and Adaptation," Working Conference on Complex and Dynamic Systems Architecture, 2001.
[11] Jaesun Kim, Sooyong Park, "Goal-based Evaluation of Contextual Situations for Self-adaptive Software,[10] The Korean Institute of Information Scientists and Engineers, Vol.33, No.3, pp. 316-334, 2006.
[12] Thomas Buchmann, Alexander Dotor, and Bernhard Westfechtel, "MODEL-DRIVEN DEVELOPMENT OF GRAPHICAL TOOLS: FUJABA MEETS GMF," In Proceedings of the 2nd International Conference on Software.

이 준 훈

2008년 2월 성균관대학교 정보통신공학부(공학사). 2008년 3월~현재 성균관대학교 대학원 전자전기컴퓨터공학과 석사과정. 관심분야는 소프트웨어공학, 오토노믹 컴퓨팅, 성능 평가



박 정 민

2008년 2월 한국산업기술대학교 컴퓨터공학과(공학사). 2005년 2월 성균관대학교 대학원 컴퓨터공학과(공학석사). 2005년 3월~현재 성균관대학교 대학원 컴퓨터공학과 박사과정. 2008년 3월~현재 동양공업전문대학 전임강사. 관심분야는 소프트웨어공학, 오토노믹 컴퓨팅, 자가치유 소프트웨어, 컴포넌트 기반 개발 방법론

이 은 석

1985년 2월 성균관대학교 전자공학과(공학사). 1988년 3월 일본 동북(Tohoku)대학교 대학원 정보공학과(공학석사). 1992년 3월 일본 동북(Tohoku)대학교 대학원 정보공학과(공학박사). 1992년~1994년 일본 미쯔비시 정보전자연구소 특별연구원. 1994년~1995년 일본 동북(Tohoku)대학교 Assistant Prof. 1995년 3월~현재 성균관대학교 정보통신공학부 교수. 관심분야는 소프트웨어공학, 오토노믹/유비쿼터스 컴퓨팅, 에이전트 지향 지능형 시스템 등