자가 적응형 소프트웨어를 위한 아키텍처 기반 소프트웨어 진단 기법

김규래 김동선 박수용
서강대학교
{charmian82, darkrsw, sypark }@sogang.ac.kr

Architecture-based Software Diagnosis Method for Self-Managed Software

Kyurai Kim Dongsun Kim Sooyoung Park
Sogang University

요 약

소프트웨어가 해결해야 할 문제가 점점 복잡해지고 있음과 동시에 소프트웨어의 자체의 복잡도 또한 증가하고 있다. 또한 소프트웨어 개발 시간에 예상하지 못했던 실행 환경에 노출되는 경우가 빈번해 졌다. 이러한 요구상항과 함께 소프트웨어의 유지보수와 개발을 쉽게 하기 위해 자가 적응형 소프트웨어에 대한 필요가 늘어나고 있다. 자가 적응형 소프트웨어란 실행 환경과 내부 상황을 판단하여 적절한 기능을 수행할 수 있도록 스스로 재구성 할 수 있는 소프트웨어이다. 소프트웨어가 향상된 기능으로 재구성을 하려면 자신의 내부 상황과 자원 소모량 등 소프트웨어 실행 환경에 대한 계속 적인 관찰이 필요하다. 그러나 기존의 자원 소모량 관찰에 대한 연구는 개발 시간에 프로그램 자체 효율을 위해 프로그램 개발 언어 단위에서 이루어져왔다. 예를 들면 관찰 단위가 실행시 호출되는 함수나 데이터 중심으로 진행 되어 재구성 단위인 컴포넌트 별로 이해하기가 쉽지 않았다. 따라서 본 논문은 재구성 단위를 컴포넌트로 정의 하고 메소드 단위의 호출이 생길 때마다 발생하는 데이터를 컴포넌트 단위로 추상화 시키는 기법을 제안한다.

1. 서 론

소프트웨어는 과거에 비해 점점 더 다양한 기능을 자율적으로 수행하도록 요구 받고 있다[1]. 사용자는 소프트웨어가 복잡한 문제를 해결하며, 사용자의 요구사항을 반영하여 빠르고 정확하게 작동하길 요구한다. 예를 들면 홈 서비스 로봇은 집 이라는 한정된 장소에서 일어나는 다양한 상황에 대하여 적절하게 판단, 대처하여 사용자의 요구사항을 만족시킬수 있어야 한다. 이러한 요구에 맞추어 새로운 소프트웨어 개발 패러다임으로 자가 적응형 소프트웨어가 대두되고 있다. 자가 적응형 소프트웨어란 외부의 환경이나 소프트웨어 자신의 상황변화를 관찰하여 소프트웨어 자신의 기능을 향상 시킬 수 있도록 재시작 없이 재구성을 할 수 있는 소프트웨어이다. 자가 적응형 소프트웨어는 개발 당시 예측하지 못했던 상황이나, 기존에 기능에서 새로운 기능의 추가 변경이 용이한 등의 장점을 가지고 있다[2].

자가 적응형 소프트웨어는 재구성을 하기 위해 자신의 내부 상황과 실행 환경을 계속적으로 관찰하고 있어야 한다. 소프트웨어의 내부 상황 관찰은 자신의 상태나 변수 등 소프트웨어가 스스로 변경 가능한 부분의 관찰이다. 소프트웨어 외부 관찰에는 메모리, CPU, 통신 데이터 등의 자원 소모량 관찰과 사용자의 요구사항 등 소프트웨어가 동작하는 환경에 대한 관찰이 모두 포함 된다. 예를 들면 홈 서비스 로봇은 현재 자신이 수행할 수 있는 일을 목록이나 성능에 관한 내부 상황 관찰을 하고 있어야 하며 동시에 사용자의 요구 사항, 자신의 현재 위치 등 외부 환경과 CPU, 메모리 등 자신의 계산 능력에 관계된 자원 소모량

관찰을 하고 있어야 한다. 이렇게 수집된 정보를 기반으로 로봇은 자신이 수행할 수 있는 서비스와 품질 정도를 결정하여 사용자의 요구 사항에 부합하는 아키텍처를 결정 실행시간에 동적으로 재구성 하여 요구 사항을 수행할 수 있다.

본 논문은 소프트웨어 외부 관찰 중에서 아키텍처 기반의 소프트웨어 자원 소모량 관찰 기법에 제안에 초점을 맞추고 있다. 기존 소프트웨어 자원 소모량에 관한 연구는 함수나 데이터 중심으로 이루어져 왔다. 그러나 자가 적응형 소프트웨어의 관점에서 자원 소모량은 재구성 단위인 컴포넌트 단위 별로 의미가 있다. 본 논문에서는 기존의 소스코드에서 컴포넌트를 식별하여 메소드 단위의 정보 추출을 결합하여 컴포넌트 단위별로 자원 소모량을 측정하는 기법을 제안하고 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 관하여 간략히 소개한다. 3장에서는 아키텍처기반의 정보수집에 대하여 제안하고, 4장에서는 본 논문의 결론을 제시한다.

2. 관 련 연 구

소프트웨어 공학 분야에서는 컴포넌트 재사용을 위해 소스 코드에서 컴포넌트를 찾아내는 많은 연구들이 진행되어 왔다[3]. Software Reconnaissance 기법은 실행 흔적을 분석하여 특정 기능과 코드를 매치 시키는 것이다. 이 기법은 아키텍처 관점이 아니라 단순히 하나의 기능과 그것을 수행하는 소스 코드만을 연결해 준다는 점에서 제안한 컴포넌트 기반 데이터 식별 기술과 다르다[4].

RIPPLES는 실행시 메소드 호출이 발생할때 생기는 데이터들을 수집하여, 사용자가 지정한 컴포넌트로 매치 시켜 관련 데이터를 2D로 모여주는 기법이다. 이 기법은 아키텍처의 개념이 들어 간 것이 아니라 단순히 사용자의 판단에 따라서 소스코드가 컴포넌트화 되며 커넥터에 관한 개념은 연구되지 않았다[5].

표 1 컴포넌트 식별에 대한 관련연구 비교

	컴포넌트 식별 방법	자원 소모량 보고
Reconnaissance	기능 단위로 식별	x
RIPPLES	사용자 지정	x
Dr. Bohnet	코드구성 단위에서 자동 추출	x
제안하는 기법	ADL 로 기술	o

Bohnet의 분석방법은 소스코드 상의 코드 구성 단위 (e.g. C++/JAVA: namespace/package keyword) 에서 자동적으로 컴포넌트 이름을 추출한다. 이에 따른 적합성 유무를 사용자의 의견을 반영하여 아키텍처 구조를 완성한다. 사용자의 반응에 따라서 아키텍처 구성이 달라 질 수 있다는 점에서 본 논문이 제안하는 기법과 다르다.[4]

본 논문에서는 사용자에게 의존하지 않고 ADL을 사용하여 정형화된 컴포넌트 식별 방안을 제안한다.

3. 접 근 방 법

본 장에서는 컴포넌트와 소스코드를 매치시키는지 설명하겠다. 먼저 아키텍처 모델에서의 컴포넌드에 관해 설명을 하겠다. 다음으로 제안한 기법에 대한 설명을 하겠다.

소프트웨어의 외부 자원 소모량 관찰은 소프트웨어 내의 메소드의 호출이 있을 때마다 호출에 관련된 데이터를 실시간으로 조합하는 방법을 택한다.[6] 그러나 메소드 단위의 데이터는 명시적으로 이해하기 힘든 낮은 단위의 데이터이다. 따라서 본 논문에서는 메소드 단위의 데이터를 추상화 시켜 자가 적응형 소프트웨어의 컴포넌트 단위로의 매치 방법을 제안한다.

3.1 아키텍처 모델

소프트웨어는 아키텍처는 소프트웨어를 구성하고 있는 컴포넌트와 커넥터로 이루어져 있다.[7] 컴포넌트의 일반적인 정의는 시스템이나 서브 시스템의 구조적 구성 요소로 독립적인 단위이다. 또한 잘 정의된 인터페이스를 가지고 있으며 독립적으로 실행 가능한 단위이다.[7] 본 논문에서 정의하고 있는 컴포넌트는 일반적인 컴포넌트의 정의를 확장한 개념이다. 본 논문에서는 컴포넌트를 일반적인 정의와 더불어 재구성이 가능한 최소 단위로 정의 한다

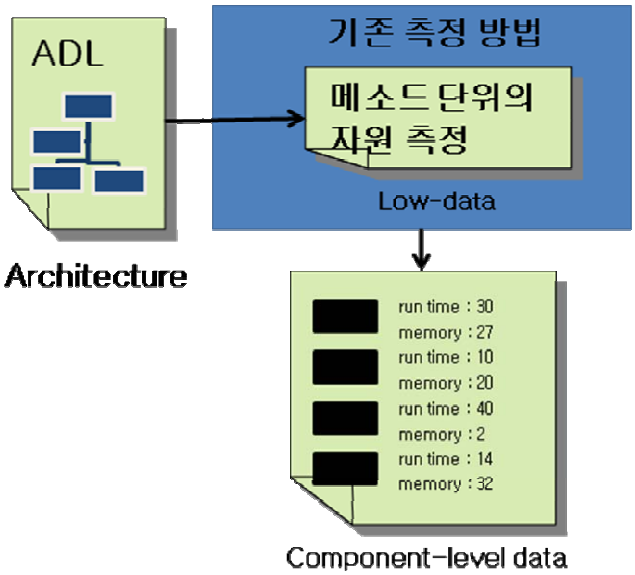

그림 1제안한 기법의 전체적인 모습

3.2 제안 기법

프로그래밍 언어 마다 특징이 다르기 때문에 본 논문에서는 프로그래밍 언어를 객체지향 언어로 한정한다. 그러나 어떠한 객체지향 언어에서도 명시적으로 컴포넌트란 타입을 지원하지 않는다.[4] 따라서 제안하는 방법은 소프트웨어의 설계상의 컴포넌트에 소스 코드 상의 구성요소(e.g. JAVA에서의 package)를 매치시켜 컴포넌트로 인식하는 방법을 제안하고 있다. 또한 이렇게 매치된 소스코드상의 구성요소 하위 레벨의 단위(e.g. JAVA: class)는 상위에 매치된 컴포넌트의 구성 요소가 된다. 이렇게 컴포넌트와 소스코드와의 관계가 성립되면 메소드 호출시에 측정되는 데이터들은 자신이 속한 컴포넌트의 자원 소모량이 된다. 예를 들어, 그림 2의 왼쪽 편에 있는 것은 홈 서비스 로봇의 부분적인 아키텍처이다. 홈 서비스 로봇은 사용자의 요구 사항을 분석하여 적절한 컴포넌트들을 호출해주는 'task_manager,' 앞으로 움직이는 'automove', 팔을 움직이는 'arm', 물체를 인식하는 'object_recognition _high', 'object_recognition_low' 컴포넌트를 가지고 있다. 먼저 'task_manager' 컴포넌트가 사용자의 요구사항을 분석하여 해당하는 컴포넌트를 호출하게 된다. 'task_manager'는 'automove', 'arm', 'Face_recognition', 'vision'컴포넌트를 호출할 수 있다. 이에 구현은 오른쪽과 같은 구성을 가지게 된다. 'automove'컴포넌트는 'Navigation'과 'Wheel' 클래스로 이루어져 있다. 만약 'task_manager'가 'automove'를 호출할 경우
클래스 Navigation과 Wheel의 소모 자원은 메소드 단위로 측정하여 'automove'의 자원 사용량으로 계산이

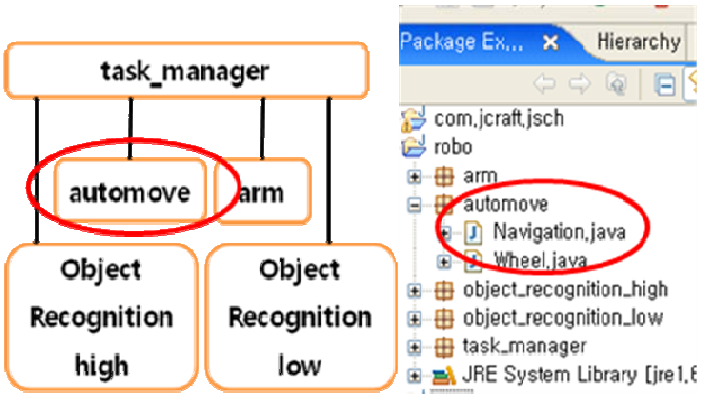

그림 2 ADL로 기술된 아키텍처 명세서와 소스코드구성요소와의 매치

된다.

만약 로봇이 주행을 하면서 물체를 인식하여 팔을 움직여 잡아야 하는 상황을 생각해 보자. 로봇은 'automove', 'arm', 'object_recognition_high'컴포넌트를 사용하고 있다. 이때 로봇이 로봇내부의 자원이 부족함을 인식한다면 'object_recognition_high'자원이 좀 덜 필요한 'object_recognition_low' 바꾸어 재구성을 하여야 한다. 이때 로봇에게 필요한 자원은 재구성 가능한 컴포넌트 단위별 정보이다. 물체인식 고품질 내의 어떠한 함수가 많은 자원을 소모한다는 것은 의미가 없다.

4. 결 론

자가 적응형 소프트웨어는 개발 시간에 예상하지 못했던 실행 환경이나 자신의 내부 상태에 따라 좀더 좋은 기능과 구조를 스스로를 변경할 수 있는 소프트웨어이다. 이러한 소프트웨어에서는 좀더 나은 기능과 구조로 위해서 자신의 내부, 외부 환경을 관찰하고 있는 것이 중요하다. 본 논문에서는 외부상황 관찰 중 컴포넌트별 자원 소모량 측정에 초점을 맞추고 있다. 아키텍처 설계상의 컴포넌트와 소스를 매치시켜 매치된 소스코드에서 소모하는 자원은 컴포넌트에서 사용하는 자원으로 처리하는 기법을 제안하였다. 제안된 방법은 자원 소모량을 재구성 단위인 컴포넌트 별로 측정을 하는것과 같은 효과를 보여 재구성 판단에 효율적인 근거가 된다.

6. 참 고 문 헌

[1] R. Laddaga, "Active Software", Paul Robertson. Howard E. Shrobe, Robert Laddaga (Eds): Self-Adaptive Software, First

International Workshop, IWSAS 2000, Oxford, UK, April 17-19, 2000.

[2] D. Garlan, S. Cheng, A.C. Huang, B. Schmerl, P. Steenkiste, "Rainbow: Architecture-Based Self-Adaptation with Reusable Infrasturcture", IEEE Computer, Vol. 37, No.10, pp. 46-54.

[3] J. Bohnet, J.Doollner, "Analyzing Feature Implementation by Visual Exploration of Architecturally-Embedded Call-Graphs", 28th Int'l Conf. on Software Engineering, Workshop on Dynamic Analysis, Shanghai, China, May 2006, pp. 41-48. 2006.

[4] N. Wilde, J. Gomez, T. Gust, D. Strasburg, Locating user functionality in old code. In Int'l Conf. on Software Maintenance, 200-205, 1992.

[5] K. Chen, V. Rajich, RIPPLES: tool for change in legacy software. In Proc. of the IEEE Int'l Conf. on Software Maintenance, 230-239, 2001.

[6] www.uml.org.