서비스 품질 기반의 아키텍처 동적 재구성*

정창해, 김동선, 박수용
서강대학교 컴퓨터학
{zenlover^{O}, darkrsw, sypark}@sogang.ac.kr

Dynamic Reconfiguration based on Quality of Services

Changhae Jung^{O} Dongsun Kim, Sooyoung Park
The Department of Computer Science, Sogang University

요 약

소프트웨어 환경의 빈번한 변화 및 사용자의 요구사항의 변화로 인하여 소프트웨어 시스템은 과거의 정형화된 환경과는 다른 동적이고 연속적, 비결정적인 비정형화된 환경에 대응할 수 있어야 한다. 특히, 환경 및 사용자의 요구사항의 변화로 인하여 소프트웨어 시스템이 제공하는 서비스에 다른 품질 속성이 요구될 경우에는 이에 소프트웨어 시스템은 이에 대응할 수 있어야 한다. 따라서 본 연구는 환경 및 사용자의 요구사항의 변화로 인한 소프트웨어 시스템의 서비스의 품질속성의 변화에 대응할 수 있는 아키텍처 레벨의 동적 재구성 프레임 워크를 제안한다.

1. 서 론

현대의 소프트웨어 시스템은 다양한 상황과 환경의 변화에 직면하였다. 과거의 소프트웨어 시스템에 대한 환경은 정형화되고, 모든 입력값들이 관찰이 가능하였으며, 또한 결정적, 정적, 이산적이었다. 따라서 환경의 변화에 매우 둔감하였다. 하지만, 현재의 환경이 이와 반대로 복잡하고, 비정형적이고, 어떤 입력값은 숨겨져 있으며 동적이고 연속적이다.

임베디드 소프트웨어 시스템의 경우에는 주변의 환경과 밀접한 관계를 가진다. 임베디드 소프트웨어 시스템은 실시간으로 특정 서비스를 제공한다. 특정 서비스를 제공하는데 있어서 같은 서비스라 하더라도 환경에서 요구하는 특정 서비스에 대한 품질속성은 틀려진다. 이에 따라서, 임베디드 소프트웨어 시스템은 현재 환경이 요구하는 특정 품질 속성을 만족 시키는 서비스를 제공하는 것이 요구되고 있다.

소프트웨어 시스템이 빈번한 환경의 변화에 인한 서비스의 품질 속성의 변화를 인식하여 적절한 품질의 서비스를 제공하는 것이 필요하다.

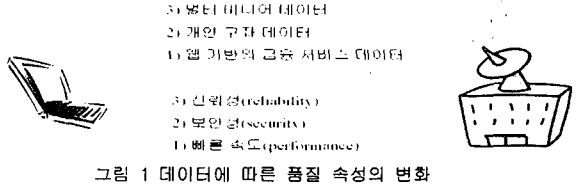

그림 1 데이터에 따른 품질 속성의 변화

예를 들면, [그림 1]과 같이 어떤 임베디드 소프트웨어 시스템이 데이터를 전송하는 서비스를 제공한다고 하자. 하지만 환경에 따라서, 또는 전송되는 데이터에 따라서 소프트웨어 시스템에 요구되는 품질속성은 틀려진다. 일반적으로 전송되는 데이터가 웹 기반의 금융 서비스 정보일 때, 그 서비스는 빠른 속도를 요구하게 된다. 주변 환경이 바뀌어서 전송되는 데이터가 개인적 은행 구좌와 관련된 정보일 경우, 보안성이 요구되며, 실시간 미디어 파일인 경우에는 소프트웨어 시스템은 신뢰성이 요구된다.

소프트웨어 시스템이 제공하는 서비스의 품질 속성은 소프트웨어 시스템의 아키텍처와 연관이 깊다[1]. 서비스의 품질 속성의 변화에 적절하게 대처하기 위해서는 소프트웨어 시스템의 구조가 바뀌어야 한다.

1990년대 이후, 자기 적응형 소프트웨어에 대한 본격적인 연구가 시작되었다. 그 이전에도 이에 대한 연구가 수행되었으나, 단지 실시간 동안에 소프트웨어의 재부팅없이 소프트웨어의 업그레이드 및 기능 변경에 초점을 둔 연구였다. 허나, 1990년대에 접어들면서, 소프트웨어가 환경을 인지하고 이에 대한 아키텍처 레벨의 동적 재구성의 관점에서 접근하기 시작하였다. Richard N. Taylor (university of California, Irvine)[2][3], David Garlan (Carnegie Mellon University)[4][5], Jamie Hillman (Lancaster University[6][7]의 아키텍처 레벨의 동적 재구성에 대한 연구가 있다. 하지만 이러한 연구들은 소프트웨어의 기능적인 부분에 대한 동적 재구성만을 다룬다.

본 연구는 아키텍처 서비스의 품질 속성을 기반으로 하는 아키텍처 레벨의 동적 재구성을 위한 프레임 워크를 제안한다. 본 연구에서 제안하는 프레임 워크는 특정 서비스에 대해서 환경의 변화에 따른 품질 속성의 변화를 소프트웨어 시스템이 반영하여 구조가 동적으로 변하도록 지원한다.

* 이 연구(논문)는 산업자원부 지원으로 수행하는 21세기 프론티어 연구개발사업(인간기능 생활지원 지능로봇 기술개발사업)의 일환으로 수행되었습니다.

2. 본 론

2.1 동적 재구성 프레임 워크 기술 요소

2.1.1 포트 기반의 아키텍처 (Port-based Architecture)

포트기반의 아키텍처[8]는 서로 연결되는 태스크들로 구성되어 있다. 각각의 태스크는 0개 이상의 입력과 0개 이상의 출력을 가지는 실행의 흐름이다. 여기서 포트(Port)는 입력과 출력이 일어나는 지점(point)을 의미한다. 컴포넌트는 0개 이상의 입력 포트와 출력 포트를 가지고 있으며, 이들 사이의 연결은 입력 포트와 출력 포트를 통해서 이루어진다. [그림2]는 포트 기반의 아키텍처의 예이다.

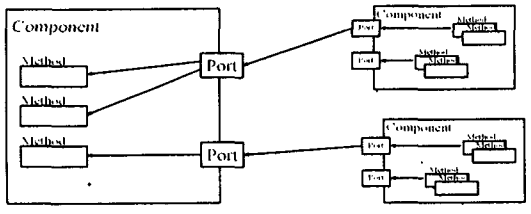

그림 2 포트 기반의 아키텍처의 예

2.1.2 동적 컴포넌트 로딩 (Dynamic Component Loading)

동적 컴포넌트를 로딩하는 방법으로는 자바에서의 리플렉션(Reflection)[9] 방법을 사용한다. 리플렉션은 형(class)를 복원할 수 없는 객체가 존재할 때 형확인을 하고, 객체를 생성하며, 멤버 메소드까지 호출할 수 있는 프로그램의 기법으로, 자바에서는 기본 라이브러리를 제공한다. 로컬 컴포넌트 및 원격 컴포넌트의 로딩도 리플레션을 이용하면 가능하다.

2.1.3 아키텍처 기술 언어 및 파서 (Architecture Description Language and Parser)

XML을 이용하여 아키텍처 및 컴포넌트의 기술 언어를 생성한다. [그림3]와 같이 아키텍처는 컴포넌트와 연결로 이루어져 있고, 컴포넌트는 이름, 경로, 설명, 입력포트 및 출력 포트, 그리고 통신에 관련된 메소드로 이루어져 있다. 연결은 통신의 흐름을 나타내는데, 통신을 보내는 컴포넌트의 포트와 받는 컴포넌트의 포트에 대한 정보를 가지고 있다.

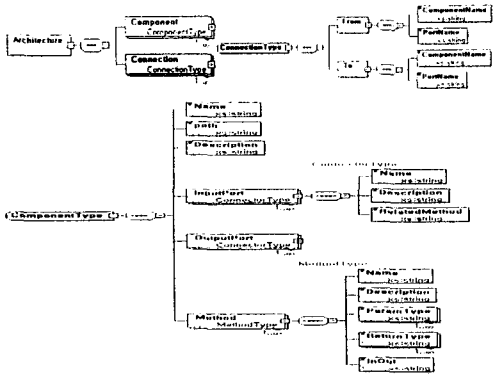

그림 3 XML 기반의 아키텍처 기술 언어

2.1.4 컴포넌트 간의 통신 방법 (Communication Mechanism)

컴포넌트 사이의 통신 방법은 포트를 통해서 스트링(String) 형식의 메시지를 보내서 서로간에 통신을 한다. 모든 컴포넌트들은 통신을 위한 메시지는 지정된 저장소로 보내는 메소드와 메시지를 받는 메소드를 가지고 있다.

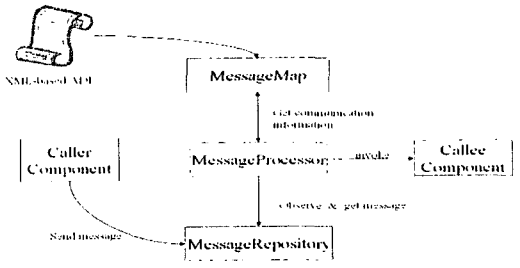

그림 4 컴포넌트 메세지 전달 메커니즘

[그림4]는 컴포넌트간의 메시지 전달 메커니즘을 나타낸다. 아키텍처 기술 언어를 통해서 필요한 메시지의 정보들을 메시지맵이 저장하고 있다. 컴포넌트가 다른 컴포넌트와 통신을 위해서 메시지를 메시지 저장소로 보내면, 메시지 프로세서가 메시지를 해당 컴포넌트로 전달하여 준다.

2.1.5 아키텍처 기술 언어를 이용한 소프트웨어 시스템의 구축

2.1.4에 있는 컴포넌트 메시지 전달 메커니즘을 이용하여 아키텍처 기술 언어의 정보를 이용하여 소프트웨어 시스템을 구축할 수 있다. XML 기반의 아키텍처 기술 언어로부터 정보를 얻어온다. 얻어온 정보에서 필요한 컴포넌트 정보를 추출해서 동적으로 로딩한다. 컴포넌트의 동적 로딩 후, 컴포넌트간의 통신에 필요한 연결 정보를 추출해서 연결시키고 컨트롤 컴포넌트를 시작시킨다.

2.2 전체적인 동적 재구성의 흐름

아키텍처 레벨의 동적 재구성을 위한 기본적인 가정은 다음과 같다.

- 환경 및 사용자의 요구사항의 변화는 인지되어 필요한 서비스의 적절한 품질 속성은 판단되어서 입력된다.
- 서비스 정보 저장소 및 컴포넌트 저장소에는 필요한 정보가 구축되어 있으며, 서비스에 대한 품질 속성에 적절한 아키텍처의 구조는 앞에서 이야기한 아키텍처 기술 언어로 표현되어 있다.

동적 재구성을 위한 전체적인 흐름은 [그림5]와 같다.

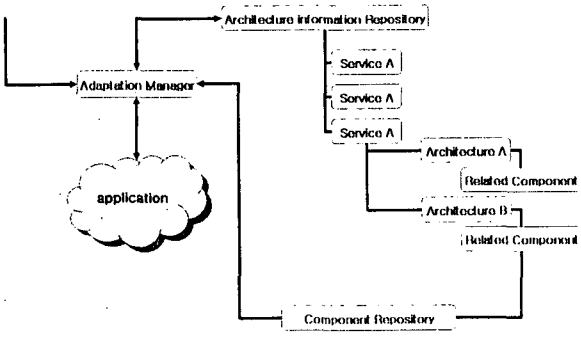

그림 5 전체적인 재구성 흐름

재구성을 위한 서비스의 품질 속성은 판단되어서 적응 관리자(Adaptation Manager)에 들어온다. 적응 관리자는 아키텍처 정보 저장소에서 해당하는 후보 아키텍처의 구조를 가져오고, 지금 현재 어플리케이션의 아키텍처 정보와 비교를 한다. 비교를 통해서 새로이 필요한 컴포넌트와 연결 정보를 얻는다. 필요한 컴포넌트는 리플렉션을 이용하여 동적 로딩을 하고, 연결을 이용하여 메시지맵을 업데이트하여 더 이상 쓰이지 않는 기존의 컴포넌트 및 연결을 제거한다.

2.3 적응 관리자(Adaptation Manager)의 구조

[그림6]은 동적 재구성을 위한 적응 관리자의 구조는 나타낸다. Architecture Discovery 컴포넌트 는 환경 또는 사용자의 요구로부터 서비스의 적절한 품질속성의 정보를 획득한다. 획득된 정보를 이용하여 아키텍처 정보 저장소로부터 후보 아키텍처 정보를 얻어온다.

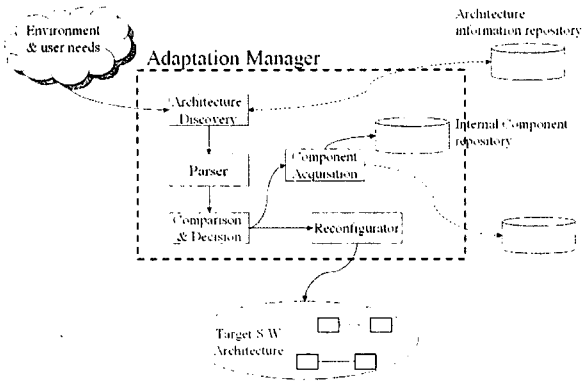

그림 6 적응 관리자의 구조

Parser 컴포넌트는 아키텍처 발견자에서 얻어온 후보 아키텍처와 현재 목표 소프트웨어 아키텍처의 정보를 분석하여 Comparison&Decision 컴포넌트로 넘긴다.

Comparison&Decision 컴포넌트는 분석된 두 아키텍처의 정보를 통해서 새로 필요한 컴포넌트의 정보를 ComponentAcquisition 컴포넌트로 넘기고, 필요한 컴포넌트 및 연결에 대한 정보를 Reconfigurator 컴포넌트로 넘긴다.

ComponentAcquisition 컴포넌트는 Comparison&Decision 컴포넌트로부터 넘어온 정보를 이용하여 필요한 컴포넌트를 지정된 내부 컴포넌트 저장소로 동적으로 로딩한다.

Reconfigurator 컴포넌트는 넘어온 정보를 이용하여 필요한 컴포넌트 및 연결을·목표 소프트웨어 아키텍처에 추가하고 더 이상 사용되지 않는 컴포넌트 및 연결을 제거하여 동적 재구성을 한다.

3. 결 론 및 향후 연구

주위의 환경의 변화와 사용자의 요구사항은 빈번하게 변한다. 그런 변화에 민감한 임베디드 소프트웨어 시스템의 경우에는 그러한 변화에 적절히 대응하는 것이 중요하다.

본 연구는 서비스 기반의 품질속성의 변화에 따른 아키텍처의 동적 재구성 프레임 워크를 제안하였다. 또한 본 연구는 동적 재구성 프레임 워크를 지원하기 위해서 컴포넌트간의 통신 메커니즘도 제안하였다. 본 연구에서 제안한 아키텍처 동적 재구성 프레임 워크를 통해서 아키텍처의 구조가 실시간에 동적으로 바뀌는 것을 간단한 테스트 프로그램을 적용시켜 확인할 수 있었다. 품질 속성을 속도 향상으로 가정하고 테스트 프로그램을 돌린 결과 동적 재구성을 통해서 속도가 향상됨을 알 수 있었다.

향후 연구는 본 동적 재구성 프레임 워크를 이용하여 실제로 복잡한 프로그램에 적용시키는 것이다. 그 외, 동적 재구성 프레임 워크를 확장시켜 주위의 환경을 관찰하여 그 정보를 획득하는 부분 및 그 정보를 분석하여 적응이 필요한 시점 및 방법을 학습하는 부분에 대한 연구가 필요하다.

4. 참고문헌

[1] Lawrence Chung, Brian A. Nixon, Eric Yu, "An Approach to Building Quality into Software Architecture", IBM Centre for Advanced Studies Conference, Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research 1995

[2] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimbingner, G. Johnson, N. Medvidovid, A. Quilici, D. S. Rosenblum, and A. L. Wolf. An architecture-based approach to self-adaptive software. IEEE Intelligent Systems, 14(3):54{62, May 1999.

[3] P. Oreizy, N. Medvidovic, and R. N. Taylor. Architecture-based runtime software evolution. In Proceedings of the 20th international conference on Software engineering, 1998.

[4] D. Garlan, S.-W. Cheng, A. C. Huang, B. Schmerl, and P. Steenkiste. Rainbow: Architecture-based self-adaptation with reusable infrastructure. IEEE Computer, 37(10):46{54, October 2004.

[5] D. Garlan and B. R. Schmerl. Model-based adaptation for self-healing systems. In Proceedings of the ACM SIGSOFT 2004 Workshop on Self-Managing Systems, 2004.

[6] J. Hillman and I. Warren. Meta-adaptation in autonomic systems. In 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, 2004.

[7] J. Hillman and I. Warren. An open framework for dynamic reconfiguration. In 26th International Conference on Software Engineering, 2004.

[8] Kevin R. Dixon, Theodore Q. Pham, Pradeep K. Khosla, "Port-Based Adaptable Agent Architecture," Paul Robertson, Howard E. Shrobe, Robert Laddaga (Eds.): Self-Adaptive Software, First International Workshop, IWSAS 2000, Oxford, UK, April 17-19, 2000

[9] ttp://java.sun.com/docs/books/tutorial/index.html