지능형 로봇의 적응성을 위한 확장된 웹 서비스*

김진한O 이창호 박동민 이병정
서울시립대학교 컴퓨터과학부
{kimjinhanO, leechangho, parkdongmin, bjlee}@venus.uos.ac.kr

An Extended Web Service for Adaptive Behavior of Intelligent Robots

Jinhan KimO, Changho Lee, Dongmin Park, Byungjeong Lee
School of Computer Science, University of Seoul

요 약

최근 소프트웨어의 생명 주기가 점점 짧아지고 있으며 시장의 요구에 수용하면서 빠르게 변화 하도록 소프트웨어를 만들어야 할 것이다. 하지만 변화가 생길 때 마다 새로운 소프트웨어를 만든다는 것은 많은 작업을 요구하는 일이 될 것이기 때문에 소프트웨어에 적응성은 중요한 이슈이다. 본 논문에서는 로봇에 초점을 맞추어 로봇이 적응성을 가지기 위해 표준 기술인 웹 서비스를 사용한다. 웹 서비스의 전형적인 표준 발견 기술은 UDDI 레지스트리에 의존하며 서비스를 찾을 때 기초적인 정보만을 사용하기 때문에 원하는 기능의 서비스를 찾는 것이 어렵다는 단점을 가지고 있다. 그래서 본 논문에서는 웹 서비스를 통해 로봇이 요구하는 컴포넌트를 발견하기 위해 기존의 UDDI를 변경하지 않고 확장된 검색을 지원하는 발견기법을 제안하고 이를 구현한 EWS를 소개한다.

1. 서 론

컴퓨터의 발전 속도는 빠르게 진행되고 있다. 날마다 새로운 제품들과 새로운 기술들이 나오고 있다. 따라서 소프트웨어도 생명 주기가 점점 짧아지고 있으며 시장의 요구에 수용하면서 빠르게 변화를 하도록 만들어야 할 것이다. 하지만 변화가 생길 때 마다 새로운 소프트웨어를 만든다는 것은 많은 작업을 요구하는 일이 될 것이다. 그렇기 때문에 소프트웨어에 있어서 적응성은 중요한 이슈가 되고 있다. 적응성이란 동작 환경의 변화에 스스로 대처하여 자신의 행위를 수정할 수 있는 능력을 말한다 [1]. 따라서 소프트웨어에 적응성은 점점 더 필수적인 능력이 되고 있다. 이 적응성을 본 논문에서는 로봇에 초점을 맞추고 있다. 로봇이 적응성을 가지기 위해 XML 기반의 표준 기술인 웹 서비스를 사용한다. 현재 표준 웹 서비스 발견 기술들은 전형적으로 UDDI 레지스트리에 의존하고 있다. 이는 서비스를 찾을 때에는 기초적인 정보만을 사용하기 때문에 원하는 기능의 서비스를 찾는 것이 어렵다는 단점을 가지고 있다 [2]. 그래서 본 논문에서는 웹 서비스를 통해 로봇이 요구하는 컴포넌트를 발견하기 위해 기존의 UDDI를 변경하지 않고 확장된 검색 [3]을 지원하는 발견 기법을 제안하고 이를 구현한 EWS (Extended Web Service)를 소개한다. EWS는 일반적인 웹 서비스인 데이터 서비스와 컴포넌트 제공을 목적으로 하는 컴포넌트 서비스들을 발견 또는 등록할 수 있도록 API를 제공한다. 그리고 컴포넌트 서비스를 빠르게 접근하기 위해 EWS에 컴포넌트를 직접 서비스할 수 있도록 웹 서비스를 포함 했다.

2. 관련 연구

2.1 컴포넌트 서비스

적응형 소프트웨어를 위한 서비스는 데이터 서비스와 컴포넌트 서비스 두 가지로 분류한다 [4]. 데이터 서비스는 날씨 등과 같은 그때그때 실시간 데이터를 얻는 일반적인 웹 서비스를 말한다. 컴포넌트 서비스는 로봇의 기능이나 행동에 관련된 서비스로서 짧은 시간 안에 로봇이 서비스 받을 수 있도록 UDDI 외부에 서비스에 대한 추가정보와 컴포넌트를 업로드하여 이용한다. 컴포넌트 서비스인 경우에는 SOAP 메시지에 컴포넌트를 첨부하여 전송하는데 적응형 소프트웨어는 이 컴포넌트를 CDG (Component Description Generator) [4]를 통해

* 본 연구는 한국과학재단 특정기초연구(R01-2006-000-11150-0) 지원으로 수행되었음.

필요로 하는 컴포넌트 설명을 추출한다. 컴포넌트를 가져올 때에는 되도록 소프트웨어의 작업을 줄이기 위해 컴파일 된 파일을 가져오도록 하지만 그럴 수 없는 경우에는 소스 파일을 가져와 로봇에 적합하게 컴파일하여 사용한다.

2.2 컴포넌트 기반의 적응성을 위한 아키텍처

적응성을 가진 소프트웨어는 기존의 소프트웨어에 비해 좀 더 안전하고, 좀 더 신뢰성이 있고, 비용에 있어서 효과적인 컴퓨터 시스템을 보증할 수 있다. 이러한 소프트웨어는 크게 네 가지의 목표를 가지고 있다. 첫 째는 소프트웨어가 환경에서 변화를 지원하거나 다른 소프트웨어의 목표들을 만족하기 위하여 자동적으로 스스로 재조정할 수 있어야 한다는 것 (Self-configuration)이다. 다음으로 결함이 일어났을 때 효과적으로 회복하고 결함을 식별하고 가능하다면 그것을 수리해야 한다는 것 (Self-healing)이다. 그리고 알려진 최적의 조건에 대해 현재의 수행을 평가할 수 있고, 개선을 시도하기 위해 정책들을 정의한다는 것 (Self-optimization)이다. 마지막으로는 우발적이거나 외부의 공격들로부터 자신을 방어해야 한다는 것 (Self-protection)이다. 따라서 이런 목표들을 달성하기 위해서 소프트웨어는 자신의 내부 상태를 알고 있어야 하고, 현재 외부에 작용하고 있는 조건들을 알고 있어야 하며, 환경들이 변화하는 것을 감지해서 적당히 적응할 수 있어야 한다 [5]. 이런 소프트웨어는 대부분 크게 세가지 모듈로 이루어져 있다. 그것은 그림 1과 같이 모니터, 결정, 재구성이다.

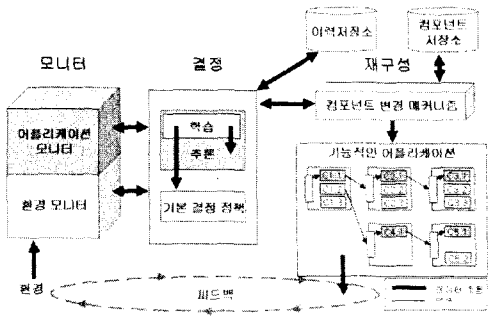

그림 1. 적응형 소프트웨어의 아키텍처

모니터 (Monitor)는 센서를 통해 얻은 정보로 외부 환경의 변화를 감지한다. 그래서 그 감지한 변화를 결정 (Decision)에 넘겨주면 여기에서는 기본 정책과 학습과 추론을 통해 적응을 위한 적절한 컴포넌트를 찾게 된다. 그러면 재구성 (Reconfiguration)에서는 그 컴포넌트로 아키텍처를 재구성하게 된다 [1, 6]. 이 상황을 다시 모니터에서 변화로 인식해서 그 결과에 대해서 판단을 하게 된다. 이러한 과정의 반복을 통해 소프트웨어는 적응성을 수행한다. 하지만 소프트웨어에 있어서 적응성이 오랜 시간 일어나게 될 때 적응성의 방향이 소프트웨어의 목적에 맞지 않을 수도 있다. 이를 위해 추가로 이력저장소 (History Repository)를 두어 적응성을 위해 아키텍처가 변할 때 마다 상태 정보들을 저장하게 한다. 그래서 위와같이 목적에 맞지 않을때 쉽게 예전의 구성으로 되돌릴 수 있는 롤백 (rollback)메서드를 제공할 수 있는 근거를 제공한다. 그럼으로써 주기적으로 소프트웨어의 목적을 점검하여 소프트웨어의 신뢰성을 유지하게 한다 [7].

3. 웹 서비스 발견 아키텍처

본 논문에서는 지식 공유 및 증식을 위한 소프트웨어의 예로 로봇에 중점을 둔다. 이 로봇은 지식과 기능을 통합하고 이의 공유와 증식을 위해서 블랙보드 [8]를 사용하고 있다. 로봇이 작동 중에 적응성을 위해서 웹 서비스를 요청할 때 짧은 시간 안에 로봇이 서비스 받을 수 있도록 서비스 제공자는 EWS에 서비스 등록과 함께 관련 컴포넌트를 업로드 한다. EWS는 서비스 등록과 검색을 도와주며 컴포넌트를 위한 별도의 저장소를 가지고 있다.

로봇의 내부는 크게 두 부분으로 이루어져 있다. 첫번째 부분은 센서를 통해 환경의 변화를 감지하고, 그 정보를 바탕으로 적응성을 위해 추론이 이루어지며, 그 결론에 따라 재구성이 이루어지는 부분이다. 두번째 부분은 내부에서 해결할 수 없는 문제가 발생했을 때, 웹 서비스를 이용하여 원하는 서비스를 검색하고 찾은 서비스를 가져오는 부분이 있다.

EWS는 기본적으로 UDDI를 변경하기 않고 외부에 추가 정보들을 저장하여 웹 서비스를 찾는데 도움을 주고 있다. 그림 2처럼 EWS는 추가적인 정보를 가지고 UDDI의 검색을 도와주는 하나의 웹 서비스이다. 이 서비스는 서비스를 등록하고 검색하는 API를 가지고 있다. 이 API들은 UDDI에서의 서비스를 등록하고 검색해주는 API를 확장한 것이다. 따라서 사용자들은 기존의 UDDI의 사용법만 알면 EWS에서 똑같이 서비스를 검색할 수 있는 장점을 가진다. 그리고 컴포넌트 서비스의 제공자인 경우

는 서비스를 등록할 때 제공하는 컴포넌트를 EWS에 있는 컴포넌트 저장소에 같이 등록해야 한다.

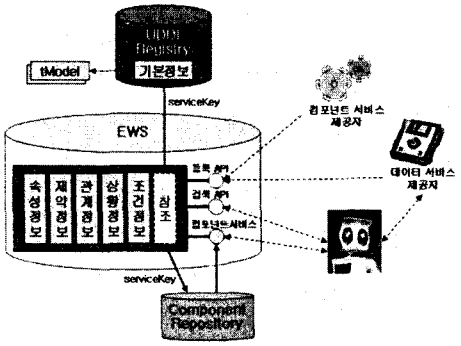

그림 2. EWS 아키텍처

EWS에서 사용되는 추가적인 정보들은 제약(constraint), 관계 (relationship), 상황 (situation), 속성 (property), 조건 (condition)정보들이다. 제약 정보는 UDDI에 등록되지 못하는 WSDL (Web Service Description Language)의 속성들을 의미한다. 관계 정보는 서비스들간의 관계를 세 가지 종류로 나타낸다. 첫 번째 관계는 상호보완적인 관계를 나타내며 서비스들 사이가 밀접하게 결합 되고, 소비자가 서비스들 중 하나를 호출하면 관련된 나머지 서비스 모두를 호출 한다. 두 번째 관계는 기능적 관계를 나타내며 하나의 서비스와 또 다른 서비스 사이에 관계는 의미상 같거나, 유사한 기능성들을 가지고 대체할 수 있는 관계를 나타낸다. 세 번째 관계는 참조적 관계를 나타내며 서비스들간의 서로 호출될지 모르는 관계를 의미하고, 서비스들간에 호출이 필수적인 상호보완적인 관계와 대조를 보인다. 속성 정보는 특정한 도메인에서 서비스 타입을 명시하기 위해 서비스 속성에 대한 스키마를 가지며 서비스들의 비 기능적 특징들을 정의한 집합이다. 상황 정보는 문맥에 따라 적절한 서비스를 선택하기 위한 정보를 가지고 있다. 그리고 조건 정보는 로봇 내부에서 적응성을 위해 추론을 할 때 사용하는 선 조건 (pre-condition)과 후 조건 (post-condition)정보로 활용한다. 조건 정보를 사용하면 로봇 내부에서 필요로 하는 기능에 대해서 추론이 일어날 때 사용하는 정보를 그대로 서비스 검색에 사용을 하기 때문에 가져오는 서비스에 대한 신뢰도를 높여주게 된다. 따라서 원하는 기능에 대한 서비스 검색이 가능하게 된다.

```
<businessService serviceKey="16CA02" businessKey="A54DC3">
<!-- EWS 에 등록 -->
    <properties serviceKey="16CA02">
        <property>
            <name>response</name>
            <responseTime>00:00:00.500</responseTime>
        </property>
    </properties>
    <relationships serviceKey="16CA02">
        <relationship type="Complementary">31AD09<relationship>
    </relationship>
    <constraints serviceKey="16CA02">
        <constraint type="bind"><style>rpc</style></constraint>
    </constraints>
    <situations serviceKey="16CA02">
        <infrared>
            <ID>H04</ID>
            <model>SE2006A</model>
            <refereshTime>00:00:00.200</refereshTime>
            <value>354</value>
        </infrared>
    </situations>
    <conditions serviceKey="16CA02">
        <Pre-condition>10</Pre-condition>
        <Post-condition>50</Post-condition>
    </conditions>

<!-- UDDI 에 등록 -->
    <name>getShortestPath</name>
    <description>A shortest path for robots</description>
    <bindingTemplates/>
</businessService>
```

그림 3. EWS의 서비스 등록 설명

그림 3은 이와 같은 추가적인 정보가 포함된 서비스 등록 설명을 보여준다. 전통적인 UDDI에 서비스에 대한 기본 정보를 등록하고 서비스의 참조를 얻어 EWS에 XML 형식으로 나머지 정보를 저장한다. EWS에서는 이러한 정보들을 가지고 보다 원하는 기능을 가진 서비스를 찾을 수 있도록 도와준다.

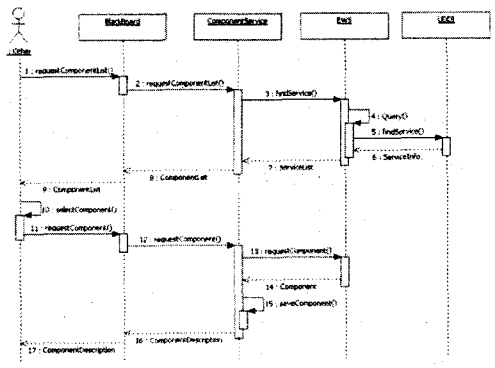

그림 4. 웹 서비스를 통한 컴포넌트 획득 순서

기본적으로 적응성은 로봇 내부에서 모니터, 결정, 재구성의 반복으로 이루어지게 된다. 로봇 내부에 컴포넌트를 위한 저장소가 있어 적응성을 위해

재구성이 이루어질 때 여기에서 필요한 컴포넌트를 가져가게 된다. 하지만 로봇 내부의 저장소의 공간은 한계가 있고 모든 상황에 대한 컴포넌트를 내부 저장소에서 가지고 있을 수는 없기 때문에 웹 서비스 같은 외부의 저장소의 필요성이 존재한다. 그리고 필요할 때만 외부의 저장소에 접근을 하여 필요한 컴포넌트만 가져오면 된다. 그림 4는 내부에서 해결할 수 없는 상황에 대해서 적응성이 일어나는 순서를 보여준다. 처음 로봇 내부에서 필요로 하는 컴포넌트에 대한 검색 정보를 CS (Component Service)에 요청하면 CS는 EWS에 등록된 추가적인 서비스 정보를 토대로 만족하는 서비스 리스트를 작성하여 로봇에게 전달한다. 이때 CS는 검색과 변환 메커니즘을 담당한다. 이후 로봇은 서비스 리스트 중 하나를 선택해서 필요한 컴포넌트를 CS에 요청하면 CS는 EWS가 제공하는 컴포넌트 서비스에 직접 접근하여 컴포넌트를 가져와 로봇이 원하는 위치에 컴포넌트와 컴포넌트 설명을 함께 저장 한다. 그리고 로봇 내부에서는 컴포넌트 설명의 정보를 기초로 자동 배포가 이루어지고 컴포넌트들의 재구성이 일어나게 된다. 컴포넌트 설명에는 로봇 내부에서 컴포넌트와 각각의 컴포넌트를 관리하는 컴포넌트 매니저에 대한 정보가 기술되어있다.

4. 구현

로봇 내부는 기본적으로 JAVA로 구현되어 있다. 로봇이 필요로 하는 컴포넌트를 얻기위해 조건들을 블랙보드에 명시하면 이벤트 핸들러에 의해 CS가 불려진다. CS는 블랙보드로 부터 조건을 얻어 웹 서비스를 통해 EWS에 서비스 리스트를 요청하고 그 결과를 블랙보드에 명시하는 기능과 원하는 컴포넌트를 EWS를 통해 내려받아 서비스 설명에 명시된 위치에 컴포넌트를 저장하는 기능을 한다.

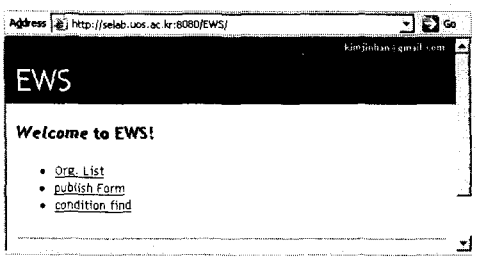

그림 5. EWS의 메인 화면

EWS는 UDDI와 통신하기위해 JAXR(Java API for XML Registries) 구현체인 Apache-Scout 패키지와 UDDI에 저장되지 못하는 정보를 보관하기 위해 데이터베이스를 가진다. 또한 컴포넌트를 직접 제공하기위해 웹서비스 형태로 컴포넌트 내려받기 서비스를 가진다. UDDI는 Apache의 jUDDI를 사용하였다.

그림 5는 jUDDI에 웹을 통해 접근할 수 있도록 하기 위하여 EWS에 JAVA의 서블릿을 이용한 웹 어플리케이션을 보여준다.

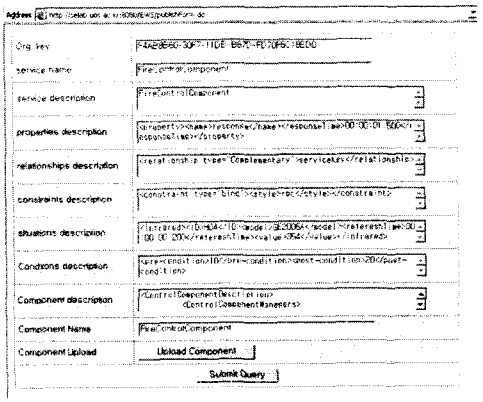

그림 6. EWS의 서비스 등록 화면

컴포넌트 서비스 제공자는 그림 6의 EWS 웹 인터페이스를 통해 서비스를 등록한다. 서비스 이름, 서비스 설명, 그리고 바인딩 정보만 jUDDI에 저장되고 검색을 위한 나머지 설명들과 컴포넌트 파일은 EWS에 저장된다.

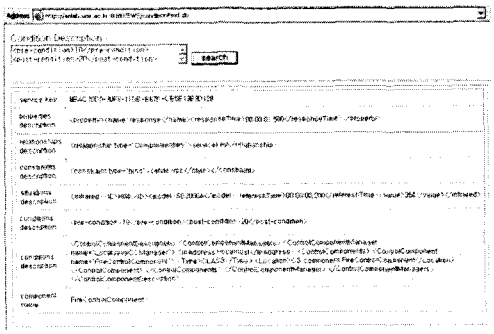

그림 7. EWS의 검색 결과

그림 7은 EWS에 저장된 컴포넌트 서비스를 조건 정보를 가지고 검색 했을 때 검색 결과를 보여주고 있다.

그림 8은 EWS을 통해 가져온 컴포넌트에서 컴포넌트 설명을 추출한 후 컴포넌트 설명에 명시된

위치의 컴포넌트를 바인딩 하고 이 컴포넌트를 제어하는 컴포넌트 매니저에 할당시킨 후 컴포넌트 매니저에 의해 실행되는 화면을 보여준다.

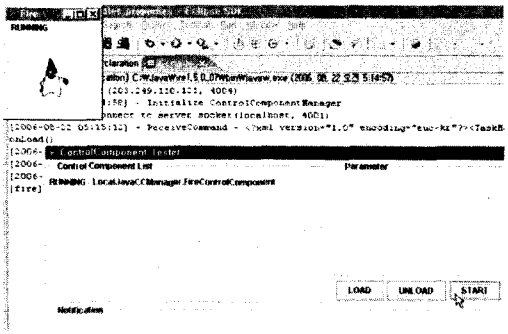

그림 8. 로봇내부로 가져온 컴포넌트의 실행

5. 결론 및 향후 연구

본 논문에서는 소프트웨어의 적응성을 위한 방법으로 웹 서비스를 통해 필요한 컴포넌트를 찾고 적용하는 방법을 제안한다. 소프트웨어 내에서 해결하지 못하는 문제가 발생했을 때에는 외부의 저장소인 UDDI를 검색하여 문제를 해결할 웹 서비스를 찾는다. EWS를 통해 서비스를 찾을 때에는 서비스 소비자가 요구하는 기능을 만족하기 위해서 속성, 관계, 제약, 상황, 조건 정보 같은 검색 정보들을 추가해서 더 나은 서비스를 찾도록 한다. 그리고 서비스 제공자들은 그 규약에 맞게 서비스를 등록하고 컴포넌트 서비스인 경우에는 컴포넌트도 EWS에 같이 등록을 해야 한다. 서비스 소비자인 로봇은 여러 검색 조건을 이용하여 찾은 서비스를 로봇 내부로 가져와 시스템에 맞는 컴포넌트로 변환된다. 그리고 변환된 컴포넌트는 아키텍처를 재구성하는데 쓰인다.

향후 연구로는 서비스의 기능에 맞는 효율적인 검색을 위해서 온톨로지를 기반으로 서비스의 기능을 의미적으로 표현하여 서비스를 찾는 연구가 진행되어야 할 것이다. 그리고 최종적으로는 다양한 지식 공유 및 증식을 위한 소프트웨어에도 적용될 수 있도록 일반화된 연구가 필요하다.

참고문헌

[1] 구형민, 박유식, 김기현, 고인영, 최호진, "아키텍처 기반의 자가 성장 로봇 소프트웨어를 위한 저장소 구조," *Proc. of Korea Conference on Software Engineering*, pp. 219 - 227, 2006.

[2] D. Rocco, J. Caverlee, L. Liu and T. Critchlow, " Domain-specific Web Service Discovery with Service Class Description," *Proc. of the IEEE International Conference on Web Services*, 2005.

[3] J. Liu, N. Gu, Y. Zong, Z. Ding, S. Zhang and Q. Zhang, " Service Registration and Discovery in a Domain-Oriented UDDI Registry," *Proc. of The 5th International Conference on Computer and Information Technology*, 2005.

[4] 김진한, 이창호, 이광우, 이병정, 김희천, "적응형 소프트웨어에 대한 웹 서비스 발견 기반 기법," 한국컴퓨터종합학술대회 논문집, 제 33 권, 제 1(C)호, pp. 175-177, July 2006.

[5] M. G. Hinchey and R. Sterritt, "Self-Managing Software," Computer, vol. 39, Iss. 2, pp. 107 – 109, Feb. 2006.

[6] A. Diaconescu and J. Murphy, "A Framework for Using Component Redundancy for self-Optimising and self-Healing Component Based System," *Proc. of WADS workshop in International Conference Science Engineering*, 2003.

[7] J. C. Georgas, A. van der Hoek and R. N. Taylor, " Architectural Runtime Configuration Management in Support of Dependable Self-Adaptive Software," *Proc. of the workshop on Architecting dependable systems*, 2005

[8] H. V. Brussel, R. Moreas, A. Zaatri and M. Nuttin, " A behaviour-based blackboard architecture for mobile robots," *IECON '98. Proceedings of the 24th Annual Conference of the IEEE*, vol. 4, pp. 2162-2167, Aug. 1998.