적응형 소프트웨어에 대한 웹 서비스 발견 기반 기법

김진한°, 이창호, 이광우, 이병정, 김희천*
서울시립대학교 컴퓨터과학부, 한국방송통신대학교 컴퓨터과학과*
{ kimjinhan°, leechangho, freeism, bjlee}@venus.uos.ac.kr, hckim@knou.ac.kr*

A Web Service Discovery based Approach to Adaptive Software

Jinhan Kim°, Changho Lee, Kwangwoo Lee, Byungjeong Lee, Heechern Kim*
School of Computer Science, University of Seoul,
Dept. of Computer Science, Korea National Open University*

요 약

소프트웨어는 많은 상황을 염두에 두고 만들어진다. 하지만 모든 문제를 해결하도록 만들 수는 없기 때문에 문제가 발생했을 때 그 문제를 해결할 수 있도록 적응성을 두어야 한다. 본 논문에서 제시하는 아키텍처는 WSAS, 컴포넌트 변환, 적응성을 위한 부분으로 구성된다. WSAS는 기존의 UDDI를 변경하지 않고 확장된 검색을 지원하는 서비스이다. 그리고 컴포넌트 서비스를 위한 컴포넌트 저장소를 제공한다. 변환 아키텍처는 검색된 서비스를 통해 컴포넌트를 변환하고 변환된 컴포넌트는 적응형 아키텍처에서 재구성을 통해 문제를 해결한다. 본 논문에서는 컴포넌트 기반의 아키텍처에 적응성을 위해 개선된 웹 서비스 발견 기법을 제안한다.

1. 서 론

적응성을 가진 소프트웨어란 "동작 환경의 변화에 스스로 대처하여 자신의 행위를 수정할 수 있는 소프트웨어"를 말한다. 적응성을 해결하기 위해서 주로 소프트웨어는 외부에서 해결책을 가져오는 방법에 있다 [1].

웹 서비스는 인터넷 기반의 프로토콜을 통해 XML 기반의 메시지들을 사용하여 다른 소프트웨어 에이전트들과 함께 직접적인 상호작용들을 지원하는 기술이다 [2]. 따라서 여러 다른 언어와 플랫폼에서 구현된 것과 상관없이 사용할 수 있다는 장점이 있다. 그렇지만 웹 서비스를 컴포넌트 기반의 소프트웨어 개발방법(Component Based Software Development: CBSD)의 환경으로 가져와 사용하도록 제시된 방법은 없었다.

본 논문에서는 적응성을 가지는 소프트웨어를 위해 웹 서비스를 발견하는 기법과 발견한 웹 서비스를 컴포넌트로 변환하는 방법을 제안한다.

2. 관련 연구

2.1 웹 서비스 발견 기술

표준 웹 서비스 발견 기술들은 전형적으로 UDDI 레지스트리에 달려있다. 이는 서비스 이름, 설명, 카테고리, 그리고 바인딩 정보와 같은 초보적인 서비스 정보만을 저장한다. 서비스 소비자들은 키워드와 카테고리만을 통해 서비스 발견을 시도하므로 찾고자 하는 기능의 서비스를 찾는 것이 어렵다는 단점을 가지고 있다 [3]. 그래서 서비스 소비자들이 요구하는 기능을 만족하기 위한 발견기법들이 제안됐으며, 크게 UDDI를 변경하는 기법 [4]과 UDDI를 변경하지 않고, 추가적인 정보를 외부에 저장하는 기법 [5]으로 나누어 볼 수 있다.

첫 번째 방법은 서비스 발견을 위한 검색이 레지스트리 안에서 일어나기 때문에 빠르고 효과적이라는 장점이 있다. 그러나 UDDI를 변경시키기 때문에 표준 인터페이스를 무시하는 단점이 있다. 두 번째 방법은 서비스 발견을 위해 검색 조건에 속성과 제약 그리고 서비스들간의 관계를 추가하는 기법이다. 이 방법은 기존의 UDDI에 외부 데이터베이스를 연결해 놓고 이 곳에 추가 정보들을 저장해 놓는다. 속성들과 관계들을 가지고 서비스를 검색하면 결과에 대한 만족도를 높일 수 있을 뿐만 아니라 제약조건으로 한 번 더 필터링을 해서 더 나은 서비스를 찾게 해 준다. 그러나 이 방법은 로봇과 같은 임베디드 영역에 대해서 센서로부터 얻은 상황에 대한 정보들이 추가되지 않아서 서비스 소비자들이 요구하는 정확한 서비스를 찾을 수가 없다.

2.2 컴포넌트 기반의 적응성을 위한 아키텍처

적응성을 가지는 소프트웨어에서는 기본적으로 3-Tier의 아키텍처를 가지게 되는데 모니터, 결정, 재구성이 그것이다. 모니터(Monitor)는 센서를 통해 외부 환경의 변화를 감지하는 부분이다. 그래서 그 감지한 변화를 결정(Decision)에 넘겨주면 여기에서는 학습과 추론을 통해 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. 하지만 대부분의 방법에서 재구성이 소프트웨어의 목적에 맞지 않게 되었을 때 되돌아갈 수 있는 메커니즘을 제공하지 않는다.

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

3. 웹 서비스 발견 아키텍처

본 논문에서는 적응형 소프트웨어의 예로 로봇에 초점을 둔다. 그림 1은 전체적인 아키텍처를 보여준다. 서비스는 컴포넌트 서비스와 데이터 서비스 두 가지로 분류한다. 컴포넌트 서비스는 로봇의 기능이나 행동에 관련된 서비스로서 짧은 시간 안에 로봇이 서비스 받을 수 있도록 WSAS에 등록과 함께 관련 컴포넌트를 업로드 한다. 데이터 서비스는 날씨 등과 같은 그때그때 필요한 서비스들로써 일반적인 웹 서비스를 말한다. WSAS는 서비스 등록과 검색을 도와주며 컴포넌트 저장소를 가지고 있다. 그리고 로봇 내부에는 문제를 인지하고 해결 방법을 결정하는 부분과 그에 맞는 서비스를 검색하고 가져온 결과로 재구성하는 메커니즘이 있다.

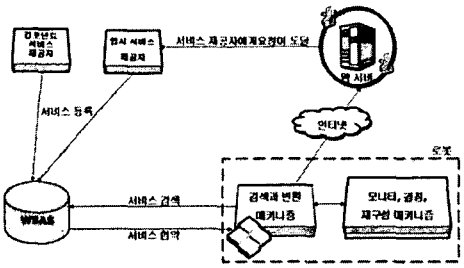

그림 1. 적응형 로봇 소프트웨어에 대한 웹 서비스 기반 기법의 아키텍처

3.1 Web Services Assistance Service (WSAS)

관련연구에서 기술했듯이 UDDI를 통한 서비스 발견은 로봇 도메인에 만족스러운 검색 결과를 가져다 주지 못한다. 특히 실행 시간 중에 로봇의 적응성을 위해 적합한 서비스 발견이 필수적이다. WSAS는 기존의 UDDI를 변경하지 않고 추가적인 서비스를 통으로써 확장된 기능을 지원한다. 이 서비스의 추가적인 정보들을 이용하기 위해서는 서비스 제공자와 서비스 소비자는 WSAS의 API의 상세한 정보를 알아야 하지만 WSAS는 더 나은 서비스를 찾는 방법을 제시 한다.

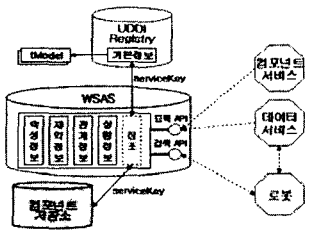

그림 2. WSAS가 더해진 아키텍처

그림 2에서 WSAS는 두 가지의 API를 가진다. 이들은 기존 UDDI와 같은 기능을 가진 API를 제공하며 추가적인 정보를 등록하고 검색하기 위해 기존 API를 확장한다. 등록 API는 UDDI에 저장이 되는 기본 정보들에 속성, 제약, 관계, 상황에 대한 정보들을 입력 받는다. WSAS에는 네 가지의 추가 정보들과 서비스에 대한 참조 정보가 저장이 되고 기본 정보는 UDDI에 저장을 한다. 서비스가 컴포넌트 서비스일 경우에는 해당 컴포넌트를 컴포넌트 저장소에 저장한다. 검색 API는 추가 정보들을 가지고 WSAS를 통해 UDDI에서 찾고자 하는 서비스들을 검색한다. 또한 검색한 서비스가 컴포넌트 서비스인 경우에는 WSAS의 컴포넌트 저장소를 통해 바로 컴포넌트를 얻는다. 그리고 기존 UDDI 방식으로 접근을 해도 WSAS의 API는 UDDI의 API를 포함하기 때문에 문제없이 접근이 가능하며 기존 UDDI에 등록 및 검색하는 효과를 준다. 그림 3은 WSAS에 등록된 서비스의 예이다. 기본 정보는 UDDI에 저장하고 추가적인 정보는 WSAS에 저장되며 이들은 "serviceKey"를 통해 서로 연결된다.

```
<businessService serviceKey="myServiceKey" >
<!-- 추가적인 정보 -->
    <properties serviceKey="myServiceKey">
        <property>
            <name>response</name>
            <responseTime>00:00:01.500</responseTime>
        </property>
    </properties>
    <relationships serviceKey="myServiceKey">
        <relationship type="Complementary">serviceKey<relationship>
    </relationship>
    <constraints serviceKey="myServiceKey">
        <constraint type="bind"><style>rpc</style></constraint>
    </constraints>
    <situations serviceKey="myServiceKey">
        <infrared>
            <ID>H04</ID>
            <model>SE2006A</model>
            <refereshTime>00:00:00.200</refereshTime>
            <value>354</value>
        </infrared>
    </situations>
<!-- UDDI 기본 정보 (UDDI 에 저장)-->
    <name>getShortestPath</name>
    <description>A shortest path for robots</description>
    <bindingTemplates/>
</businessService>
```

그림 3. WSAS 데이터 구조

추가 정보에 대한 설명은 다음과 같다:

- **속성 정보(property)**는 특정한 도메인에서 서비스 타입을 명시하기 위해, 서비스 속성에 대한 스키마를 가지며 서비스들의 비 기능적 특징들을 정의한 집합이다. 예를 들면 서비스 이용 가격, 응답 시간 등이 서비스의 속성 정보이다.
- **서비스들간의 관계 정보(relationship)**는 세가지 형태로 분류한다. 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.
- **서비스의 제약 정보(constraint)**란 UDDI에 등록되지 못하는 WSDL의 속성들을 의미한다. 이는 UDDI 검색 후 최종 서비스 제공자의 WSDL 정보를 통해 알 수 있는 입출력 파라미터, 오퍼레이션 타입, 네트워크 전송 프로토콜, 바인딩 정보들을 사전에 미리 알 수 있으며, 검색의 조건으로 사용된다.
- 로봇과 같은 임베디드 소프트웨어에 필요한 **상황 정보(situation)**는 물리적인 요소를 가지는 서비스 소비자에게 필요한 정보이다. 로봇이 서비스를 발견하려고 할 때 필요한 지식을 얻기 위하여 상황 정보를 검색의 조건으로 사용한다. 상황 정보는 로봇의 센서에 의한 데이터 또는 데이터들의 변화이다. 로봇의 센서의 종류를 분류할 수 있으나, 센서의 위치와 측정범위는 상이하다. 그러므로 로봇의 모델에 따른 센서 위치 정보를 정의하는 위치 스키마와 센서 각각의 특징을 정의하는 센서 스키마를 두어 WSAS와 서비스를 발견하는 서

비스 소비자가 참조할 수 있도록 한다.

3.2 컴포넌트 변환

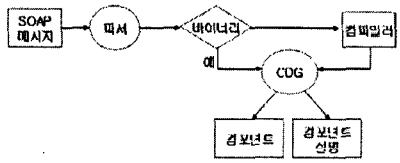

그림 4. 컴포넌트 변환 과정

컴포넌트 서비스인 경우에는 WSAS를 통해 받은 SOAP 메시지를 컴포넌트로 변환해야 한다. 그림 4는 이 과정을 보여준다. SOAP 메시지를 파서를 통해 컴포넌트 부분만 추출한다. 컴포넌트를 얻은 후엔 앞으로 구현할 CDG(Component Description Generator)를 통해 컴포넌트 설명을 생성한다. 컴포넌트를 가져올 때에는 되도록 소프트웨어의 작업을 줄이기 위해 컴파일 된 파일을 가져오도록 하지만 그럴 수 없는 경우에는 소스 파일을 가져오도록 한다.

```
<?xml version="1.0" encoding="UTF-8" ?>
<ComponentDescription>
    <Component>
        <name>Distance</name>
        <type>session</type>
        <subtype>Stateless</subtype>
        <home>DistHome</home>
        <remote>Dist</remote>
        <class>Distance</class>
        <method>
            <name>Distance</name>
            <method-intf>Remote</method-intf>
            <method-name>getDistance</method-name>
        </method>
    </Component>
</ComponentDescription>
```

그림 5. 컴포넌트 설명 예

그림 5는 컴포넌트 설명에 관한 예를 보여준다. 이 컴포넌트는 센서를 통해 거리를 구해주는 컴포넌트로 무 상태(Stateless)로 로직을 처리하고, getDistance() 메소드를 가지며, 로봇을 위해 인터페이스를 제공한다. 컴포넌트 설명은 로봇이 컴포넌트의 인터페이스에 접근하는 방법을 제공한다.

3.3 이력 관리

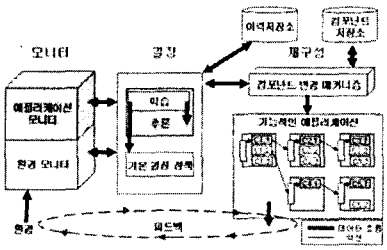

그림 6. 적응형 아키텍처

본 논문에서는 결정 부분에 초점을 둔다. 결정은 두 가지 경우에 일어난다. 대부분은 문제를 해결하지 못했을 때 일어나는 경우이지만 시스템이 유휴 상태일 때 더 나은 해결책을 찾기 위해 일어나는 경우도 있다. 위 두 경우에 검색 메커니즘에 요청을 하게 되고 변환 메커니즘을 거친 컴포넌트는 어플리케이션에 반영이 된다. 이 아키텍처에는 이력저장소(History Registry)를 두어 적응성을 위해 아키텍처가 변할 때 마다 상태 정보들을 저장하게 한다. 이 저장소는 변화의 방향이 시스템의 목표에 맞지 않는 방향으로 변경이 된 경우에 시스템의 신뢰성을 유지하고 [7] 쉽게 이전의 상태로 돌아갈 수 있는 방법을 제시하는 롤백(rollback) 매서드를 위해서 존재한다.

4. 구현

그림 7은 컴포넌트 서비스를 WSAS에 등록하는 화면이다. WSAS의 프로토 타입은 MS(Microsoft)에서 제공하는 UDDI에 접근하는 웹 어플리케이션을 확장하여 구현하였다.

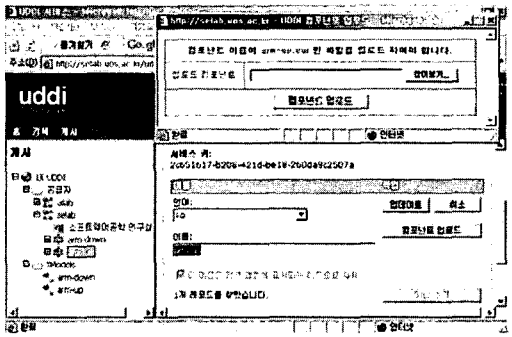

그림 7. 컴포넌트를 WSAS에 등록

5. 결론 및 향후 연구

본 논문에서는 소프트웨어의 적응성을 위한 방법으로 웹 서비스를 찾고 적용하는 방법을 제안한다. 소프트웨어 내에서 해결하지 못하는 문제가 발생했을 때 외부의 UDDI를 검색하여 문제를 해결할 웹 서비스를 찾는다. WSAS를 두어 찾을 때에는 서비스 소비자가 요구하는 기능을 만족하기 위해서 검색 정보들을 추가해서 더 나은 서비스를 찾도록 한다. 그리고 서비스 제공자들은 그 규약에 맞게 서비스를 등록해야 한다. 이렇게 찾은 서비스는 로봇 내부로 가져와서 시스템에 맞는 컴포넌트로 변환된다. 그리고 변환된 컴포넌트는 아키텍처를 재구성하는데 쓰인다.

향후 연구로는 웹 서비스 기반의 적응성을 가진 소프트웨어의 프로토 타입을 확장하는 것이다. 그리고 서비스의 기능에 맞는 효율적인 검색을 위해서 상황 정보에 관한 연구가 좀 더 필요하다. 그리고 최종적으로는 다양한 적응형 소프트웨어에도 적용될 수 있도록 일반화된 연구가 필요하다.

참고 문헌

[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] M. Endrei, J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Krogdahl, M. Luo and T. Newling, *Patterns: Service-Oriented Architecture and Web Services*, 2004.

[3] 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.

[4] S. Ran, "A Model for Web services Discovery with QoS," ACM SIGecom Exchanges, Vol. 4, Iss. 1, 2003.

[5] 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.

[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.