웹 서비스 애플리케이션의 동적 성장을 위한 ESB와 에이전트 기반 프레임워크

이 창 호[†] · 김 진 한[†] · 이 재 정[†] · 이 병 정[††]

요 약

유비쿼터스 컴퓨팅 환경에서는 이기종 플랫폼간의 상호작용과 빠른 환경의 변화에 대처할 수 있는 능력이 필요하다. 웹 서비스는 이러한 문제를 위해 이종의 분산 서비스 또는 자원들을 활용하고 조직하기 위한 방법을 제공한다. 그렇지만 서비스 요청 시, 의미 정보의 부족으로 원하는 서비스를 찾기가 어렵다. 시맨틱 웹 서비스는 의미 정보는 제공하지만 다양한 매칭에 대한 방법은 지원하지 않고 있다. 또한 웹 서비스를 이용해서 소프트웨어에 적응과 확장 능력을 제공할 순 있지만, 서비스들을 관리하고 운영하는 방법이 필요하다. 따라서 본 논문에서는 웹 서비스 애플리케이션의 동적 성장을 위해 ESB(Enterprise Service Bus)와 에이전트 기반의 프레임워크를 제안하고 유용성을 보이기 위한 프로토타입을 제시한다.

키워드 : 동적 성장, 웹 서비스, ESB, 에이전트

A Framework for Dynamic Growing of Web Service Applications based on ESB and Agent

Changho Lee[†] · Jinhan Kim[†] · Jaejeong Lee[†] · Byungjeong Lee[††]

ABSTRACT

Software adaptation may be required to interact between heterogeneous platforms and to react to rapid change of environment in ubiquitous computing. Web service provides a way to use heterogeneous and distributed services or resources to utilize and organize them. But it is not easy to retrieve appropriate services when we search services because web service lacks of semantic information. Semantic web service provides additional information of services, but it does not support a method to match them in various ways. We can adapt and extend web applications by using web service, but a method for management and administration is still needed. Therefore in this paper, we propose a framework for dynamic growing of web service applications based on ESB(Enterprise Service Bus) and agent and provide a prototype to show its usefulness.

Key Words : Dynamic growing, Web service, ESB, Agent

1. 서 론

최근 유비쿼터스 컴퓨팅 기술과 지능로봇에 대한 기술의 발달로 로봇이 주변의 기기들과 상호작용을 통해 인간 생활의 향상에 도움을 줄 시기가 다가오고 있다. 이러한 환경에서는 이기종 플랫폼간의 상호작용과 빠른 환경 변화에 대처할 수 있는 능력이 필요하다. 이러한 문제들을 해결하기 위해 최근 각광받는 기술로 웹 서비스를 언급할 수 있다. 웹 서비스는 인터넷 상에 존재하는 이종의 분산된 서비스 또는 자원을 활용하고 조직하기 위한 방법을 제공한다 [1]. 웹 서비스로 구성된 애플리케이션의 성장은 서비스 요청자가 그들의 요구사항에 만족하는 서비스 제공자들을 자동적으로 찾고, 시스템이 중단되지 않으면서 새로운 서비스로 적응하는 것이 중요하다. 그렇지만 표준 웹 서비스는 서비스 요청과 발견 시 의미정보의 부족으로 원하는 서비스를 찾기가 어렵다. 반면에 시맨틱 웹 서비스는 의미정보를 사용하여 서비스의 능력들에 대한 자동화된 매칭(matching), 서비스들의 순서화(ranking) 그리고 서비스 발견(discovery) 작업을 도와준다[2, 3]. 그렇지만 대부분의 연구들이 간단한 매칭 방법(예: exact matching)만을 제공하고 의미정보의 다양한 매칭에 대한 방법은 제시하지 않고 있다.

시스템의 규모가 커지고 환경 변화의 속도가 빨라짐에 따라 애플리케이션의 적응 능력의 필요성은 점점 커져가고 있다. 하지만 모든 상황에 대해서 적응한다는 것은 불가능하

※ 본 연구는 한국과학재단 특정기초연구(R01-2006-000-11150-0) 지원으로 수행되었음.
† 준 회 원 : 서울시립대학교 컴퓨터통계학과 석사과정
†† 정 회 원 : 서울시립대학교 컴퓨터과학부 교수(교신저자)
논문접수 : 2007년 9월 17일, 심사완료 : 2007년 12월 12일

기 때문에, 이런 경우에는 외부의 도움을 받아야 할 것이다. 외부의 도움을 받는 것은 두 가지로 생각해볼 수 있다. 첫째는 애플리케이션이 센서를 통하여 받은 정보를 판단하여 적응하는 반응 적응(reactive adaptation)이다. 둘째는 소프트웨어 개발자에 의해 나중의 필요를 예측하여 기능을 확장하는 예측 확장(proactive extension)이 있다. 두 경우 모두 애플리케이션의 정지 없이 일어나는 것이지만, 차이점은 변화가 일어나는 원인이 각각 환경과 개발자의 예측에 의해 발생한다는 것이다. 이러한 두 가지의 경우를 쉽게 도와 주는 것이 또한 웹 서비스이며, 웹 서비스와 의미적 정보를 이용한 적응에 관한 연구도 활발히 이루어지고 있다 [4, 5, 6]. 하지만 이 연구들은 적응과 확장을 위해 사용하는 웹 서비스들을 어떻게 관리하고 운영할 것인지에 대한 방법을 제시하지 못하고 있다.

웹 서비스를 운영하기 위해 사용되는 ESB (Enterprise Service Bus)[7]는 다른 애플리케이션들을 통합해 데이터의 원활한 연동을 실현하기 위한 기술인 EAI(Enterprise Application Integration)의 하나이다. ESB는 서비스와 애플리케이션 같은 자원들을 통합하고 연결하는 미들웨어 패턴이라 정의할 수 있다. 이런 ESB의 특징은 다음과 같다. 표준 기술을 기반으로 하기 때문에 위험을 최소화한다. 또한 새로운 표준들과 기존의 표준을 조합하여 사용 가능하며, 메시지와 서비스 및 이벤트 기반 아키텍처를 지원한다. 하지만 ESB는 웹 서비스를 위한 표준 환경을 제공해줄 뿐 적응을 위한 방법은 개발자가 책임을 져야 하는 한계를 가진다.

따라서 본 논문에서는 웹 서비스 애플리케이션의 적응과 확장을 위해 ESB와 에이전트 기반의 프레임워크를 제안한다. 이 프레임워크는 에이전트의 자율성, 협동성, 상호관계적인 특성을 이용하여 지능적인 서비스 검색과, ESB를 사용한 동적인 적응에 대한 방법을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 서비스 발견을 위한 요구사항에 대한 연구들과 소프트웨어를 동적으로 재구성하는 연구들을 소개한다. 3장에서는 서비스를 이용한 동적 성장 프레임워크와 성장을 위해 프레임워크에서 제안하는 방법들을 소개한다. 그리고 4장에서는 반응 적응과 예측 확장을 설명하기 위한 프로토타입을 보여주고, 5장에서 본 연구의 장단점을 포함한 토의를 기술한다. 마지막으로 6장에서는 결론과 향후 연구를 기술한다.

2. 관련 연구

2.1 서비스 발견을 위한 요구사항 분석

소프트웨어 개발 과정에 있어서 사용자들이 진정으로 무엇을 원하는 지를 기술하고 그것을 찾는 것은 매우 중요하다. 그런 면에 있어서 OMG에서 제안한 UML의 유즈케이스 모델은 사용자의 요구사항을 나타내기 위해서 많은 사람들이 사용하는 방법이다. 이런 유즈케이스 모델을 분석하여 리팩토링하고 재구성하는 연구들이 진행되었다 [8, 9, 10]. 이 연구들은 유즈케이스 모델을 분해하고, 분해된 것들을 조작하기 위한 방법이나 가이드라인을 정의하고 사용한다. 이러한 방법들은 특히 요구사항이 사용자의 의도에 맞지 않게 명세 되었을 때 유효하다. 이 연구들에서는 유즈케이스를 재구성하기 위해 확장(extension), 유사(similarity), 동등(equivalence), 선행(precedence), 포함(inclusion)과 같은 유즈케이스들 간의 관계를 정의한다 [8, 9]. 그러나 이 연구들은 요구사항을 분석하고 재구성함에 있어서 웹 서비스의 특징을 고려하지 않기 때문에, 이러한 문제를 해결하기 위해서는 다른 방법이 필요하다.

요구사항으로부터 서비스들을 식별하기 위해 유즈케이스에 태스크를 사용한 연구도 진행되었다 [11, 12]. 이 연구들은 기본적으로 서비스들을 인식하기 위해 유즈케이스를 태스크로 분해한다. 공통점은 분해한 태스크들을 통해 유즈케이스를 재구성하여 유즈케이스와 서비스들을 대응시킨다. 또한 태스크들의 유사성, 독립성 등에 중점을 두어 유즈케이스를 재구성하는 방법보다 적은 수의 관계를 정의하고 있다. 그렇지만 재구성된 유즈케이스들의 관계와 순서 등과 같은 방법들을 제시하지 않고 있다.

한편으론 시맨틱 웹 서비스를 설명하기 위해 UML의 활동도를 이용해 진행된 연구들이 있다 [13, 14]. 이 연구들은 복합적인 서비스들의 구조와 순서를 정확하게 표현하고, 웹 서비스의 입/출력을 표현하기 위해 활동도를 사용한다. 시맨틱 웹 서비스의 온톨로지 개념들을 사용하기 위해, 클래스로 변환하여 나타내고 있다. 그래서 온톨로지의 참조를 통해 활동도를 완성하고, 그에 맞는 적절한 서비스를 찾는 과정을 설명하고 있다. 그렇지만 이 연구들의 문제점은 사용자들의 요구사항을 제대로 분석하여 정확하게 나타내지 않고 개발자가 임의로 분석하여 바로 활동도로 표현한다는 것이다. 따라서 요구사항에 대한 정확한 이해의 부족으로 사용자의 의도가 잘못 반영되어 개발될 수 있다.

2.2 소프트웨어의 적응

소프트웨어의 적응은 예외적인 상황을 다루는 능력이다. 환경의 변화나 사용자 요구사항의 변화 그리고 시스템 기능의 향상 때문에 소프트웨어의 적응은 피할 수 없는 문제이다. 따라서 소프트웨어의 적응 능력은 높은 적응성과 가용성을 위해 시스템에서 점점 더 많이 요구되는 능력이다. 이러한 소프트웨어의 적응에 관한 연구는 여러 분야에서 활발하게 진행되어왔다 [15, 16, 17]. 로봇이 가진 능력을 최적화하고 증가시키기 위한 연구[15], 컴포넌트 기반 시스템의 적응을 위한 연구[17]등 다양한 분야에서 다양한 관점을 가지고 연구가 진행되었다. 그렇지만 컴포넌트 기반의 시스템 같은 특정 기술에 의존적인 적응 방법들은 다양한 기술들이 존재하는 유비쿼터스 컴퓨팅 환경에서는 적합하지 않다. 따라서 특정한 기술이나, 특정한 플랫폼에 독립적인 적응 방법이 필요하다.

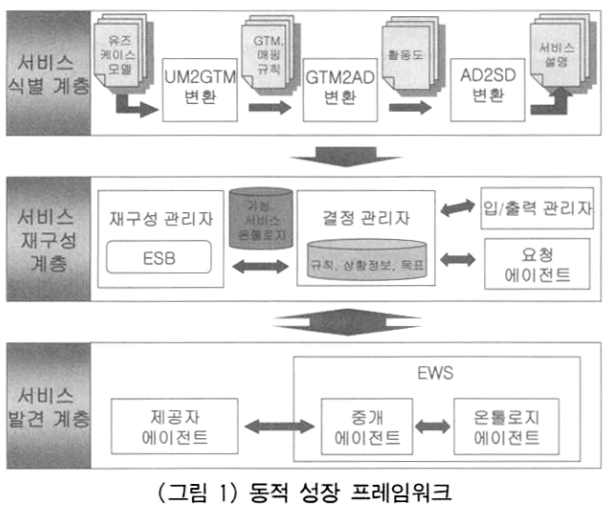
적응 능력을 가진 시스템은 환경의 변화에 대응하기 위해 스스로 기능을 재구성하고(Self-configuration), 오류가 발생

하였을 때 회복할 수 있고(Self-healing), 외부의 영향으로부터 방어할 수 있어야 한다(Self-protection) [16]. 이러한 목적을 달성하기 위해서 시스템은 스스로 내부의 상태를 인식하고, 시스템 외부 환경의 변화를 모니터링하고, 스스로 적절하게 적응할 수 있어야 한다. 이를 위한 아키텍처는 일반적으로 모니터링, 결정, 재구성의 세가지 부분으로 나누어진다. 모니터링 부분은 센서뿐만 아니라 시스템 내부의 상태를 통해 환경의 변화를 감지한다. 모니터링 정보는 결정 부분으로 보내지고 결정부분은 시스템이 어떻게 변화를 다룰지를 결정한다. 시스템의 재구성은 실제로 결정부분의 결과에 따라서 재구성 부분에서 수행된다. 이러한 연구들은 주로 컴포넌트 기반 시스템에서 진행되었다. 그러나 웹 서비스 기술의 발달에 따라, 이러한 구조를 사용한 연구들이 진행되었다. 그러나 시스템의 재구성을 위해 웹 서비스를 사용하기보다 추가적으로 웹 서비스를 이용하는 것에 초점을 두었다. 웹 서비스를 적용하는 것은 동적으로 웹 서비스를 조합 할 수 있다는 것을 의미한다. 이러한 특징은 동적 적응과 [18]의 연구에서 제안된 ESB 기반의 연구에서 적용되었다. 그러나 이러한 연구는 시스템의 재구성에 초점을 맞추고 있고 환경을 모니터링하고 결정을 내리는 기능은 불충분하다.

3. 동적 성장 프레임워크

(그림 1)은 본 연구의 동적 성장 프레임워크를 보여준다. 본 프레임워크는 서비스 식별 계층(Service Identification Layer, SI), 서비스 재구성 계층(Service Reconfiguration Layer, SR) 그리고 서비스 발견 계층(Service Discovery Layer, SD)으로 구성된다.

SI 계층은 유즈케이스 모델로부터 일련의 과정을 통해 최종적으로 서비스 설명을 생성한다. 먼저 사용자의 새로운 요구사항에 의해 작성한 유즈케이스 모델을 목표와 태스크 모델(Goal & Task Model, GTM)로 변환한다. GTM은 요구사항을 좀 더 정형적으로 그리고 웹 서비스의 속성들을 나타



(그림 1) 동적 성장 프레임워크

내주는 요구사항 분석 방법이다. 이 모델로부터 매핑 규칙을 통해 직관적인 활동도로 변환한다. 마지막으론 온톨로지 검색을 통해 온톨로지에 있는 개념들로 서비스 설명을 작성하는 것이다. 서비스 설명은 SR 계층으로 보내져 서비스 요청 시 사용된다.

SR 계층에서는 웹 서비스 애플리케이션의 재구성이 일어나는 부분이다. 재구성은 반응 적응과 예측 확장에 의해 발생한다. 예측 확장은 소프트웨어 개발자로부터 작성된 서비스 설명을 가지고 새로운 서비스를 검색하여 이루어지고, 반응 적응은 입/출력 관리자로부터 받은 정보를 바탕으로 규칙을 통해 이루어진다. 본 논문에서 제안하는 동적 성장 프레임워크는 반응 적응과 예측 확장을 가능하게 해주는 프레임워크를 말한다. 예측 확장의 경우 SI 계층으로부터 생성된 서비스 설명을 사용하여 요청 에이전트를 사용하여 웹 서비스들을 요청하고, 발견한 서비스들을 사용하여 재구성 관리자에서 웹 서비스 애플리케이션을 동적으로 재구성한다. 반응 적응의 경우에도 입/출력 관리자를 통해 받은 정보와 결정 관리자에 있는 규칙과 온톨로지 정보들을 통해 새로운 서비스가 필요한 경우에는 요청 에이전트를 통해 서비스를 요청한다.

SD 계층은 서비스의 등록과 검색을 도와주는 계층이다. 서비스 제공자들은 제공자 에이전트를 통해 서비스를 등록할 수 있다. EWS(Extended Web Service)는 표준 웹 서비스를 확장하여 향상된 웹 서비스 검색을 도와준다 [19]. EWS 내부에는 중개 에이전트와 온톨로지 에이전트가 존재한다. 중개 에이전트는 요청 에이전트와 제공자 에이전트로부터 오는 요청들을 처리한다. 또한 서비스 검색 시 온톨로지의 개념과의 타입 매칭을 통하여 적합한 서비스를 찾도록 도와준다. 온톨로지 에이전트는 도메인을 설명하는 도메인 온톨로지와 서비스를 기술하는 서비스 온톨로지로 저장한다. 온톨로지는 기본적으로 OWL(Web Ontology Language)로 표현되고, 서비스 온톨로지는 OWL-S(Web Ontology Language for Web Service)를 사용하여 표현한다.

3.1 서비스 식별

본 논문에서는 사용자의 요구사항을 나타내는 유즈케이스 모델과 활동들로 서비스를 표현하는 활동도 사이의 간격을 줄이기 위한 새로운 서비스 설명 방법을 제안한다. 이 방법은 예측 확장을 위해 사용되며 자연어보다 정형적인 방법으로 요구사항을 기술한다. 또한 이 방법은 온톨로지 에이전트에 저장된 웹 서비스들의 특징을 표현하기 위해 IOPE(input, output, precondition and effect) 정보를 사용한다. 그리고 비 기능적인 요구사항을 위해서는 QoS 정보를 포함한다. QoS 정보는 발견된 서비스 리스트로부터 이 정보를 만족하지 않는 서비스를 필터링하기 위해 중개 에이전트에 의해 사용된다.

사용자의 의도를 고려하기 위해 유즈케이스 모델을 사용하여 기술하며, 요구사항을 분해하여 목표와 태스크 모델로 재구성한다. 목표와 태스크 모델에 대한 정의는 다음과 같다.

정의 1. 목표 & 태스크 모델

Goal = <ID, Task, IOPE, QoS>

Task = <Func, IOPE, S-name, S-num, QoS>

- ID: 유즈케이스의 이름
- Func: 태스크를 식별해주며 동사와 목적어 형태를 가진다.
- IOPE: 목표와 태스크의 입/출력, 선조건 그리고 결과를 나타낸다.
- QoS: 비 기능적인 특성들 중 비용(cost)과 응답시간(response time)을 포함한다.
- S-name: 서비스 식별을 위해 필요한 서비스 이름.
- S-num: 순서에 따라 할당한다. 동시에 수행되는 경우에는 '2.a', '2.b', '2.c', 그리고 조건 분기인 경우에는 '2.1', '2.2', '2.3'와 같은 표기를 사용한다.

태스크는 유즈케이스 모델의 흐름에서 더 이상 분해될 수 없는 최소한의 단위를 말한다. 목표는 유즈케이스로 식별될 수 있는 전체 태스크들의 집합이다. 서비스는 하나 이상의 태스크들로 구성된다. 목표와 태스크의 IOPE는 전체와 부분 집합의 관계를 가진다. QoS는 웹 서비스의 비 기능적인 속성이다. 본 논문에서 비용과 응답시간을 다루는 이유는 웹 서비스 사용에 대한 비용을 지불해야 하며, 인터넷의 특성상 서비스의 응답 시간은 중요한 요소이기 때문이다. 태스크 각각의 QoS의 값의 합은 목표의 QoS의 값을 초과해서는 안 된다. S-name은 서비스를 식별하기 위해 사용되지만, 개발자의 경험과 지식에 많이 의존한다. 목표와 태스크의 항목들을 유즈케이스의 정보들로 채운 후에, 태스크들로부터 서비스를 유도하는 작업을 해야 한다. 서비스 유도를 위한 기본 가이드라인은 다음과 같다.

- 전체 대응: 전체 태스크들이 하나의 서비스로 식별되는 경우이다. 전체 대응은 전체로써 하나의 특별한 기능을 가지며, 부분적으로는 의미가 없는 경우에 사용된다.
- 부분 대응: 목표가 여러 태스크의 집합들로 이루어져있고, 각각의 집합은 특별한 기능을 가지며 서비스로 식별된다. 이 경우 서비스들간의 순서와 관계는 기술되어야만 한다.
- 재사용: 이전에 서비스로 식별되었던 태스크들이 있을 경우 사용한다.

세가지 경우로 서비스를 식별하고 남은 태스크들이 서비스로 식별되지 못하는 경우에는 개발자의 판단에 의해 태스크의 누락이 없게 한다. GTM의 S-name과 S-num은 서비스의 식별이 이루어진 후에 채워진다. 서비스 식별 과정은 서비스 검색 결과에 의해 피드백을 받아 반복적으로 이뤄져야 한다. 이 방법이 이전의 연구들과 다른 점은 분할한 태스크를 가지고 서비스를 식별하며, 서비스들간의 순서와 관계를 고려한다는 것이다.

GTM을 활동도로 변환하기 위해 매핑 규칙을 정의한다. 매핑 규칙은 다음과 같다.

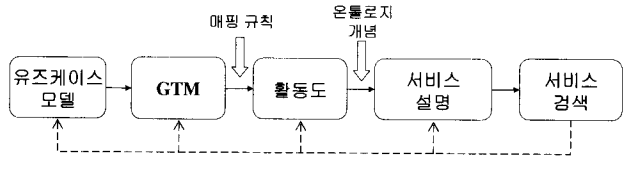
정의 2. 매핑 규칙

- S-name(태스크) à 활동의 이름
- Input(태스크 or 목표) à 활동의 입력 핀들의 집합
- Output(태스크 or 목표) à 활동의 출력 핀들의 집합
- Precondition(태스크 or 목표) à 활동으로 들어오는 감시조건(guard condition)들의 집합
- Effect(태스크 or 목표) à 활동으로부터 나가는 감시조건들의 집합
- S-num(태스크) à 활동도에서 순서
- QoS(태스크 or 목표) à 태스크의 QoS는 활동으로부터 나가는 감시조건의 집합에 기술한다. 그리고 목표의 QoS는 마지막 노드로 들어가는 감시조건에 기술한다.

S-name이 활동의 이름으로 매핑될 때, 같은 S-name을 가진 태스크들은 하나의 활동으로 매핑된다. 제어 흐름(Control flow)은 활동들이 생성된 후에 S-num에 따라 더해진다. QoS는 OCL (object constraint language) 형태를 사용하여 기술한다. 태스크 각각의 QoS 정보는 활동으로부터 나가는 감시조건에 기술한다. 목표의 QoS 정보는 마지막 노드에 들어오는 감시조건에 기술한다.

(그림 2)는 서비스를 식별하는 전체 과정을 보여주는 그림이다. 활동도는 GTM 생성 후에 매핑 규칙을 사용하여 목적 시스템의 구성과 흐름을 보여주기 위해 만들어진다. 이 활동도의 정보는 서비스 설명의 정보로 추출이 되는데, 이 과정에서 시맨틱 웹 서비스의 발견을 위해 온톨로지의 개념으로 대체하여 서비스 설명을 채우게 된다. 그러므로 서비스 설명은 모든 요소들에 들어가는 내용이 활동도로부터 추출하여 온톨로지 검색 후, 개념으로 대체하여 완성한다. 서비스 설명은 활동도의 활동으로부터 생성된다. 따라서 활동도의 활동들의 수만큼 서비스 설명들이 만들어지고, 서비스 검색을 요청한다. 또한 서비스를 식별하는 프로세스는 서비스 검색을 통해 피드백을 받아 서비스 식별의 정제화를 달성한다.

(그림 3)은 활동도로부터 서비스를 요청하기 위해 하나의 활동으로부터 유도된 서비스 설명의 예를 보여준다. name은 찾고자 하는 서비스의 이름을 나타낸다. 이 요소에는 활동도에 있는 활동의 이름을 사용한다. hasInput과 hasOutput은 각각 활동의 입력핀과 출력핀으로부터 채워진다. hasPrecondition과 hasResult 요소에는 각각 활동의 들어오는 감시조건과 나가는 감시조건의 내용이 들어간다. QoS에는 나가는 감시조건의 QoS 부분으로부터 채워진다. hasInput, hasOutput, hasPrecondition, hasResult는 서비스



(그림 2) 서비스 식별 프로세스

```
<service description>
  <name>InfraredApp</name>
  <serviceCategory></serviceCategory>
  <hasInput>(direction ?X)</hasInput>
  <hasOutput>(image ?Y)</hasOutput>
  <hasPrecondition>(lux 20)</hasPrecondition>
  <hasEffect>(validImage ?Y)</hasEffect>
  <QoS>(cost 120)(responseTime 5)</QoS>
  <weight>(I 2)(O 2)(P 1)(E 1)</weight>
</service>
```

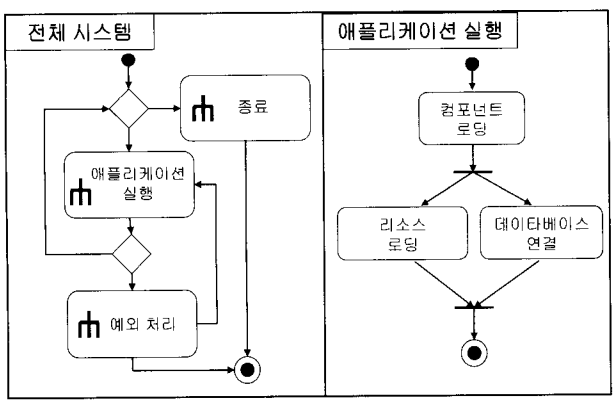
(그림 3) 서비스 설명의 예

검색 시, 서비스 매칭을 위해 후보 서비스들의 IOPE와 각각 비교를 위해 사용된다. 서비스 매칭은 가중치 값을 두어 실행하는데, 이 가중치 값들은 textDescription 요소에 표현하며 고정되어 있지 않다. 왜냐하면 가중치는 도메인에 따라 서비스 요청자의 정확한 환경 제약을 나타내기 때문이다.

3.2 활동도의 구성

활동도는 전체 시스템의 흐름을 기술하기 위한 널리 쓰이는 방법이다. 이 활동도는 유즈케이스 모델로 표현할 수 없는 제어와 자료의 흐름을 기술할 수 있다. 이런 정보의 일부는 GTM으로도 표현할 수 있지만, GTM은 전체 시스템을 직관적으로 표현하고 이해할 수 있는 표현 방법은 아니다. 따라서 활동도가 시스템을 정확하게 표현하고 직관적인 이해를 도와주기 때문에 시스템이 가지는 기능들 사이의 관계를 보여주기 위해 사용한다.

(그림 4)의 왼쪽 편의 중첩된 활동은 시스템의 구성을 보여준다. GTM으로부터 생성된 활동도는 (그림 4)의 오른쪽과 같이 시스템의 한 부분이 가지는 기능을 보여준다. 만약 활동도가 새로운 요구사항으로부터 유도되었다면, 새로운 활동이 전체 시스템의 활동도에 추가된다. 전체와 부분으로 활동도를 구성함으로써, 활동도의 복잡도를 조절하고, 요구사항 변경 또는 새로운 요구사항으로 인한 활동도 수정을 줄인다.



(그림 4) 활동도의 예

3.3 에이전트 기반 서비스 검색

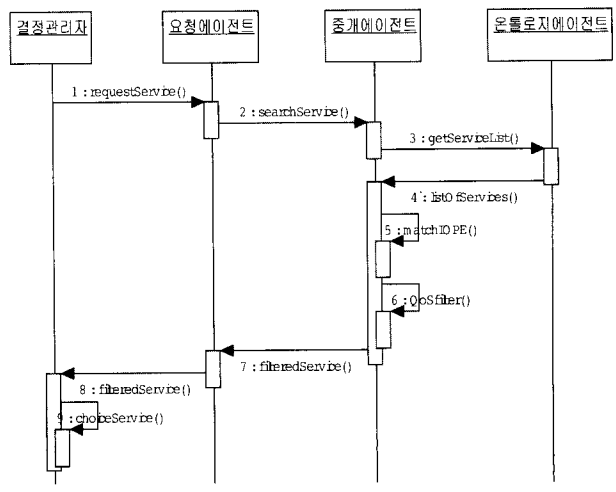
에이전트는 독립적으로 작업을 수행하는 소프트웨어이다. 대부분의 에이전트들은 직접 명령을 내리는 것 보다는 추상적인 목적을 명세하는 기법에 기반한다. 일반적으로 에이전트의 특징은 다른 에이전트와 구별할 수 있는 고유한 식별성(Identity)을 가지고, 주변 환경으로부터 정보를 받아들이고 그것에 관하여 추론을 하며, 추론을 바탕으로 자신의 행동을 수정할 수 있다. 그리고 에이전트는 다른 에이전트들과 통신할 수 있는 능력을 가지고 있다. 따라서 본 프레임워크의 에이전트를 이용한 온톨로지 기반 서비스 검색은 추론을 기반으로 하는 지능적인 검색을 가능하게 한다.

또한 향상된 서비스 검색을 위해 서비스 설명에 serviceCategory와 weight 필드를 사용한다. serviceCategory는 제한된 범위에서 검색이 진행되도록 서비스 기능에 대한 카테고리를 레이블화한 정보를 가진다. 특별한 기능을 가지는 서비스는 하나의 분류가 아니라 여러 분류에 속할 수 있다. 이런 현상은 각종 기기들이 가지는 다양한 성능을 만족하는 서비스를 위해 필요하다. 예를 들어 네비게이션을 위한 서비스를 찾는다고 할 때, 이를 위한 서비스가 가지는 기능은 네비게이션 기능뿐만 아니라 mp3 플레이어, DMB 등등 다양한 추가 기능을 포함할 수 있다. 따라서 이러한 서비스는 카테고리 필드에 (네비게이션, mp3, DMB)와 같은 정보를 포함한다.

weight 필드는 사용자의 문맥 정보를 표현하기 위해 사용한다. 본 논문에서 사용자의 문맥 정보는 프레임워크가 존재하는 환경에서 서비스 능력에 대한 정보로 제한한다. 따라서 서비스의 IOPE에 대하여 다양한 도메인에서 서비스 기능에 원하는 정도를 가중치로 표현한다. 차후에 문맥 정보에 대한 자세한 연구가 필요하겠지만, 정보의 단순함은 빠른 처리와 빠른 응답에 대한 기대를 만족시켜준다.

(그림 5)는 에이전트를 이용하여 서비스를 검색하는 과정을 보여준다. 서비스 검색은 결정 관리자에서 새로운 서비스가 필요하다고 판단을 했을 때 발생한다. 그에 따라 찾고자 하는 서비스의 서비스 설명을 요청 에이전트로 보내 서비스를 요청한다.

요청 에이전트는 시스템 외부에 존재하는 중개 에이전트



(그림 5) 에이전트 기반의 서비스 검색

로 서비스 설명을 보내 서비스 검색을 요청한다. 서비스 검색에서 중개 에이전트는 핵심 역할을 한다. 중개 에이전트는 서비스 설명을 가지고 온톨로지 에이전트로부터 서비스를 검색하여 후보 서비스들의 온톨로지 개념들을 사용하여 서비스의 IOPE에 대해서 타입매칭을 한 후, 적합한 서비스들의 리스트를 만들어준다. 이 리스트는 요청 에이전트를 통해 결정 관리자로 보내지면, 결정 관리자는 가장 적합한 서비스를 선택한다. 일반적으론 이 리스트는 매칭 함수에 의해 계산된 점수에 의해 정렬되고, 결정 관리자는 최상위에 있는 서비스를 최적의 서비스로 인식하고 그 서비스를 선택한다. 하지만 서비스 검색에 의해 넘어온 이 결과가 적절한 서비스들이 아닐 수도 있기 때문에, 소프트웨어 개발자가 개입할 여지는 남겨두어야 한다.

```
Match(request){
  matchlist = empty
  ontologyMatch(matchlist, request)
  categoryFilter(matchlist, request)
  iopeMatch(matchlist, request)
  QoSFilter(matchlist, request)
}
```

(그림 6) 서비스 매칭 알고리즘

(그림 6)은 동적 성장 프레임워크에서 제안하는 서비스 검색 시, 서비스와 서비스 설명 사이의 매칭의 정도를 지원하는 알고리즘을 보여준다. 매칭의 정도는 Exact, Subsume, Relaxed, Fail 네 가지로 분류한다. IOPE 매칭은 서비스의 기능적인 검색을 가능하게 해주고, QoS 필터링은 비기능적인 속성으로 서비스 검색을 가능하게 해준다. 이 알고리즘의 결과로는 적합한 서비스들의 리스트가 반환된다.

매칭 함수는 iopeMatch 함수를 호출하는데, 이 함수는 내부의 degreeMatch 함수를 사용하여 서비스의 IOPE의 타입과 서비스 설명의 hasInput, hasOutput, hasPrecondition, hasResult 요소들 과의 매칭 정도를 계산한다. 이 때 서비스들의 순서화를 위해 요청 정보로부터 넘어온 가중치 값을 사용한다[20]. QoSFilter 함수는 iopeMatch 함수로부터 반환된 서비스들의 결과를 가지고 요청 정보의 QoS를 만족하지 않는 서비스들을 리스트로부터 제거한다. 최종 결과로 생성된 리스트에는 적합한 서비스들이 가중치에 의해 계산된 결과에 의해 내림차순으로 정렬된다.

3.4 웹 서비스 애플리케이션의 재구성

동적 성장 프레임워크에서 웹 서비스 애플리케이션을 재구성 하기 위해 중요한 역할을 담당한 부분이 결정 관리자와 재구성 관리자이다. 이 절에서는 반응 적응에 중점을 두어 설명한다.

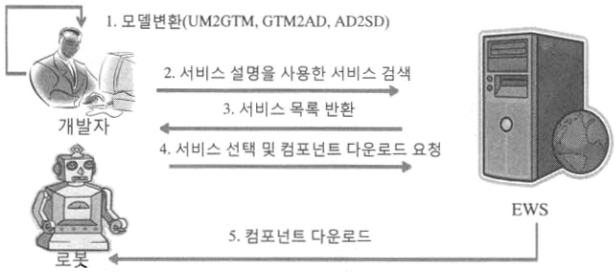
결정 관리자는 환경 변화에 적응 여부를 결정해야 한다. 이러한 결정을 내리기 위해 결정 관리자는 규칙(rule), 상황정보(context), 목표(goal) 세 가지의 정보를 필요로 한다. 목표는 웹 서비스 애플리케이션이 초기에 설정한 값으로, 애플리케이션이 생명주기 동안에 꾸준히 유지하고 달성하고자 하는 상태들에 대한 속성을 기술한다. 상황정보는 이 상태들의 현재 값을 기술한다. 규칙은 상황정보를 바탕으로 현재 애플리케이션의 상태가 목표 값과 비교하여 현재 상태를 판단할 수 있게 해주고, 그에 대처하기 위한 명령을 기술한다. 따라서 결정 관리자는 입/출력 관리자로부터 넘어온 정보를 바탕으로 내부에 목표, 상황정보, 규칙을 가지고 현재 상태에 대한 판단을 내릴 수가 있다. 판단을 통해 적응을 해야 한다면, 결정 관리자는 그에 맞는 계획을 수립한다. 이러한 계획은 환경의 변화에 대처하기 위한 내부 구조의 변경에 대한 내용이 된다. 계획은 애플리케이션의 구성을 보여주는 기능 온톨로지와 서비스 온톨로지를 가지고 구성된다. 기능 온톨로지는 애플리케이션이 지니는 기능들과 기능들간의 관계를 표현하고, 서비스 온톨로지는 각각의 기능들에 대응하는 서비스들을 기술하는 온톨로지이다.

결정 관리자는 규칙을 적용하기 위한 기능에 속하는 서비스들을 검색하여 새로운 조합으로 문제의 해결을 시도한다. 내부의 서비스들만으로 문제가 해결이 되지 않을 경우에는 원하는 서비스에 대해 서비스 설명을 생성하고, 이 정보로 검색을 통해 새로운 서비스를 찾는다.

재구성 관리자는 결정 관리자에 의해서 생성된 계획에 따라 재구성 요청을 받았을 때에, 웹 서비스 애플리케이션의 재구성을 실행한다. 재구성 관리자는 조합된 웹 서비스를 구성하고 실행하기 위해 ESB를 사용한다. ESB를 통해서 웹 서비스의 배포, 실행, 중지와 같은 명령을 수행한다. 또한 재구성 관리자는 ESB를 통해 각 서비스의 생명 주기와 종속성 정보를 가지고 있기 때문에 각 서비스의 상태를 모니터링 할 수 있다. 종속된 서비스는 재구성 관리자에 의해서 함께 관리된다.

4. 사례 연구

본 논문에서는 지능 로봇이 동적으로 성장하도록 지원하는 프레임워크의 프로토타입을 구현하였다. 지능 로봇 동적 성장 시나리오는 (그림 7)과 같다. (그림 7)은 예측 확장(순서: 1à 2 à 3 à 4 à 5)과 반응 적응(순서: 2 à 3 à 4 à 5)의 순서를 보여준다. 로봇은 다양한 기능을 가지고 있으며, 새로운 기능이 필요할 때에는 웹 서비스를 통해 필요한 컴포넌트를 찾아 다운받는다. 웹 서비스를 요청하기 위해 로봇은 내부에 요청 에이전트를 포함한다. 로봇의 요청을 처리하기 위해 외부에 동적 성장 프레임워크에서 소개했던 EWS를 구현하였다. EWS는 중개 에이전트와 온톨로지 에이전트를 포함하며, 시맨틱 웹 서비스 검색을 가능하게 해준다. 또한 로봇이 검색된 서비스의 컴포넌트를 빠르게 제공받기 위해 EWS 내부에 컴포넌트 저장소를 두어 관리한다. 프레임워크의 프로토타입은 로봇의 부재로 인해 시뮬레이터에서 테스트하였다. 프로토타입은 Glassfish V2 application



(그림 7) 지능 로봇 동적 성장 시나리오

server를 사용하여 구현하였고, 재구성 관리자의 ESB는 JBI(Java Business Integration)의 구현체인 OpenESB 2.0를 사용하여 재구성한다. EWS에 서비스를 등록하기 위하여 jUDDI를 사용한다.

4.1 반응 적응

로봇 애플리케이션의 반응 적응을 위한 시나리오는 다음과 같다.

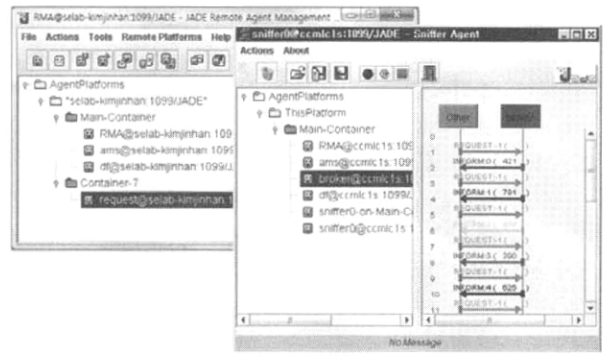
- 방 안에 있는 로봇은 비전 카메라와 적외선 카메라 모듈을 가진다. 20 lux 이상의 조도에서는 비전 카메라를 통한 정보를 처리 하는 모듈을 사용한다. 그렇지만 적외선 카메라를 사용할 수 있는 모듈은 로봇 내부에 존재하지 않는다. 따라서 전등이 켜져 있는 환경에서는 비전 카메라를 통해 시각을 바탕으로 거리를 계산하여 이동할 수 있다. 전등이 꺼져서 조도가 20 lux 이하로 낮아지면 로봇은 카메라를 통한 이동을 할 수 없다. 그러므로 로봇은 낮은 조도에서 작동할 수 있는 새로운 기능을 EWS에 요청하여 다운로드 받는다.

(그림 8)은 예제 시나리오를 반영하기 위해 구현한 시뮬레이터의 모습이다. 이 시뮬레이터는 방안의 모습을 보여주는 화면((그림 8)의 좌측)과 비전 또는 적외선 카메라의 화면(그림 8의 우측 상단), 로봇의 정보, 방안의 조명을 조절할 수 있는 버튼을 포함한다. 시뮬레이터의 사용자 인터페이스는 자바 스윙 컴포넌트로 구현하였다.

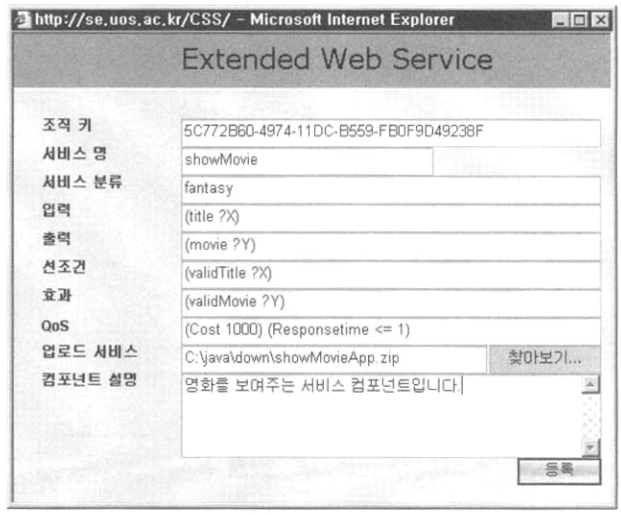
(그림 9)는 EWS의 중개 에이전트(오른쪽)와 로봇 내부에 존재하는 요청 에이전트(왼쪽)를 실행한 모습을 보여준다. 에이전트 구현을 위해서는 JADE(Java Agent Development Framework)를 사용하였다 [21]. JADE는 FIPA(Foundation for Intelligent Physical Agents)의 명세를 따르는 에이전트 애플리케이션을 개발하기 위한 자바 프레임워크이다. JADE 에이전트의 통신은 FIPA ACL(Agent Communication Language)[22] 비 동기 메시지(asynchronous message) 전달 방식에 기반한다.

(그림 10)은 EWS에 서비스의 정보들과 그 서비스가 가지는 컴포넌트의 등록화면을 보여준다. 서비스 등록 요청은 온톨로지 에이전트가 처리하며, 서비스는 jUDDI에 등록하고 컴포넌트는 컴포넌트 저장소에 저장한다.

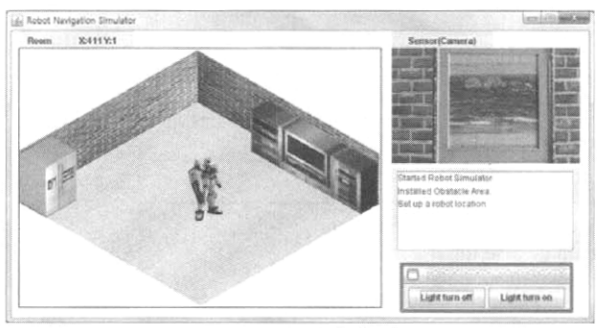
(그림 11)은 (그림 8)의 상황정보를 모델링 한 모습을 보여주고 있다. 전체 환경은 robot과 room으로 구성되고 room은



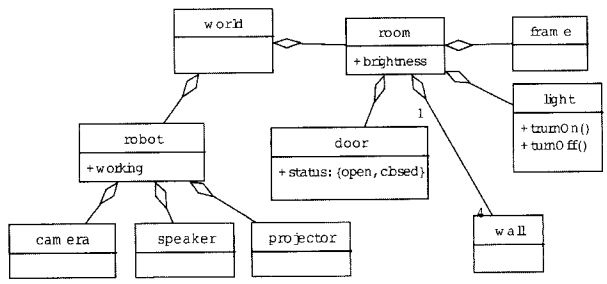
(그림 9) EWS와 로봇에서 수행중인 에이전트



(그림 10) EWS에 서비스와 컴포넌트의 등록



(그림 8) 로봇을 실행하기 위한 시뮬레이터



(그림 11) (그림 8)의 로봇 상황정보(context)

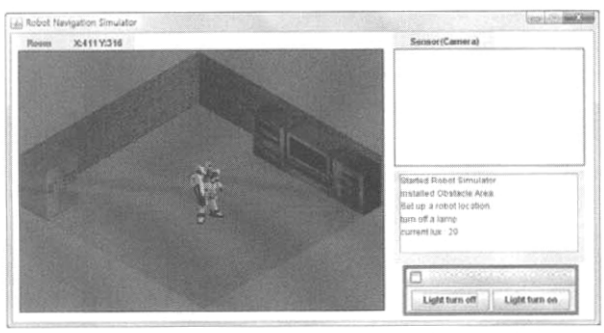
door, light, frame, 그리고 wall 으로 구성된다. 입/출력 관리자로부터 전달받은 정보는 이 상황정보의 변화이다. 이러한 변화는 규칙을 적용하여 현재 상황을 판단한다.

(그림 12)는 로봇의 결정 관리자에서 사용되는 규칙들의 일부를 Java 기반 규칙 엔진 Jess (Java Expert System Shell) 코드로 보여주고 있다. (그림 11)에서 정의한 상황모델의 요소들을 사용하여 규칙을 정의하였다. (그림 12)의 첫 번째 규칙은 로봇이 동작 중이고 조명의 밝기가 20 lux 이하 일 때, 적외선 카메라 모듈을 이용해야 된다는 것을 기술하고 있다. 두 번째 규칙은 첫 번째 규칙과는 반대로 비전 카메라 모듈을 이용해야 된다는 것을 기술하고 있다.

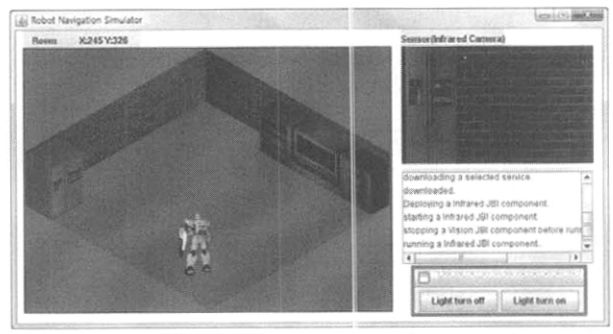
(그림 13)은 로봇이 업무를 수행하는 도중에 방안의 조명이 어두워지는 환경의 변화를 보여준다. 조명이 꺼져 조도가 20 lux 이하이면, 로봇의 비전 카메라는 작동하지 않는다. (그림 13)의 메시지 창은 방안의 조도가 20 lux로 낮아진 상태를 보여주고 있다. 이때 결정관리자는 입/출력 관리자로부터 들어오는 상황정보 변화를 입력 받아 규칙을 적용하여 현재 상황을 판단하고 기능 온톨로지와 서비스 온톨로지를 통해 필요로 하는 서비스를 검색한다. 그리고 찾은 적합한 서비스에 대응하는 컴포넌트를 다운로드 받는다.

```
(defrule infrared
   (and (robot (working on)) (room {brightness <= 20}))
   => (assert (result (value infrared))))
(defrule vision
   (and (robot (working on)) (room {brightness >  20}))
   => (assert (result (value vision))))
```

(그림 12) 로봇 규칙 예



(그림 13) 로봇 환경의 변화



(그림 14) 반응 적응 후의 로봇 시뮬레이터

(그림 14)는 재구성 관리자에서 반응 적응이 일어난 후의 모습을 보여준다. 찾은 서비스에 의해 다운로드 한 컴포넌트로 로봇이 적외선 카메라를 사용하여 낮은 조도에서도 계속 동작을 하는 것을 알 수 있다.

4.2 예측 확장

예측 확장은 미래의 필요를 예측하여 소프트웨어 개발자에 의해 기능적인 확장이 일어나는 경우이다. 본 절에서는 소프트웨어 개발자가 사용자의 여가 활용을 위한 엔터테인먼트 기능을 추가하는 과정을 보인다.

(그림 15)는 예제를 나타내기 위한 유즈케이스 모델을 보여준다. 이 모델은 하나의 성공 시나리오만 기술했지만, 원래는 사용자의 입력에 따라 달라지는 대체 흐름을 가지는 예제이다. 유즈케이스 모델로부터 GTM으로 유도 과정의 용이한 이해를 위하여 GTM까지는 성공적인 하나의 시나리오만 기술한다. (그림 15)의 유즈케이스 모델로부터 유도된 GTM은 (그림 16)과 같다.

(그림 17)은 GTM으로부터 생성된 활동도의 모습을 보여준다. 활동도는 매핑 규칙을 통해 생성되며, 이 활동도로부터 서비스 설명 정보를 추출한다. 서비스 설명은 GTM에서 식별한 서비스들의 수만큼 생성되고 그만큼의 서비스 요청이 발생한다.

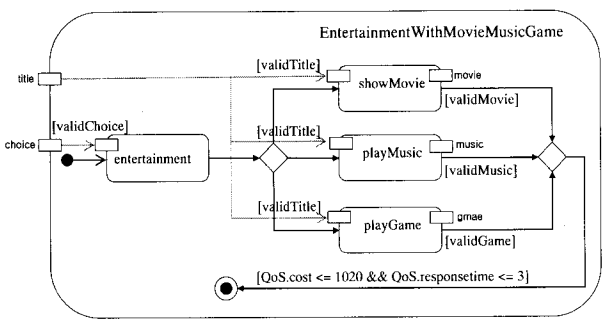
(그림 18)은 EWS를 통하여 서비스를 요청하고 그 결과를 보여주는 화면이다. 서비스 설명에 있는 정보들이 요청 화면의 항목들에 들어가고, 결과는 서비스들의 리스트로서 가장 적합한 서비스가 맨 위에 놓이게 된다. 일반적인 경우 맨 위의 서비스가 선택이 되지만, 소프트웨어 개발자가 선택하여 다운로드 될 수 있다.

Name	Entertainment
Main Actor	Robot
Pre-condition	Projector, speaker or screen is prepared.
Post-condition	A man enjoys his leisure
Successful scenario	1. A man select one between movie, music and game. 2. If he select a movie, then he also input a title. 3. Robot downloads the movie from outer repository. 4. Robot play the movie to the wall.

(그림 15) 새로운 기능에 대한 유즈케이스 모델

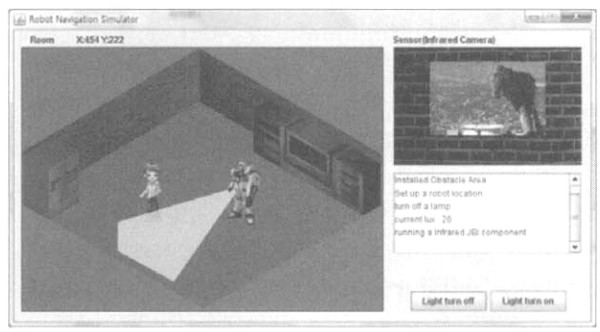
ID	EntertainmentWithMovieMusicGame						
	Func		S-name	S-num	IOPE	Cost	R-time
	Verb	Object					
Task	select	Movie, Music, Game	Entertainment	1	I: choice O: choicedProgram P: validChoice E: one of the three is progressed.	20	2
	input	Title	showMovie	2	I: title O: movie P: validTitle E: validMove		
	play	Movie	showMovie	2	I: title O: movie P: validTitle E:validMovie	1000	1
IOPE	I: choice, title O: choicedProgram, movie P: validChoice, validTitle E: one of the three is progressed, E:validMovie						
QoS	Cost <= 1020 Response time <= 3						

(그림 16) (그림 15)로부터 유도된 GTM



(그림 17) GTM으로부터 생성된 활동도

(그림 18) EWS를 통한 서비스 검색



(그림 19) 예측 확장 후의 모습

(그림 19)는 showMovie 서비스를 선택하여 그 서비스가 제공하는 컴포넌트를 다운로드하여 예측 확장을 한 로봇의 모습을 보여준다.

현재 프로토타입은 프레임워크가 가지는 기능들 중 부분적으로 구현된 상태이기 때문에 프레임워크를 제한적으로 지원하고 있다. 프레임워크의 유효성(validity)을 명확히 보이기 위해서는 다양한 상황과 시나리오를 보여주는 프로토타입이 필요하다.

5. 토 의

본 연구에서는 웹 서비스 애플리케이션의 동적 성장을 이루기 위해 에이전트들을 이용하고 있다. 동적 성장을 달성하기 위해서는 각각의 에이전트가 기능을 완벽히 수행해야 한다. 에이전트들은 웹 서비스 요청과 검색 등의 기능을 수행하고, 에이전트 내부에 추론 엔진을 가지기 때문에 검색 시, 온톨로지 개념들을 매칭한다. 하지만 적합한 서비스를 찾기 위해서는 온톨로지 정보 이외에 추가 정보들이 필요하다. 이를 위해 동적 성장 프레임워크에서는 카테고리를 레

이블화한 정보와 QoS, 가중치 정보들을 포함하여 서비스를 검색한다. 이 정보들을 사용하여 제한된 범위 내에서 검색이 가능하고, 검색 결과의 순위를 결정한다. 하지만 웹 서비스 도메인을 나타내기에 아직 QoS 정보가 부족하다. 또한 서비스 설명의 가중치는 IOPE에 대한 상대적인 중요도를 표현하지만 가중치 설정에 대한 현실적인 가이드라인은 부재하므로, 앞으로 보완해야 한다. 그리고 본 프레임워크는 반응 적응시 센서 자료로부터 서비스 설명이 적절히 유도된다는 가정을 하고 있으며, 예측 확장시 서비스 설명이 유즈케이스 모델로부터 효과적으로 유도된다고 생각되나, 유즈케이스 모델 리팩토링하기 위한 구체적인 기준이 보완되어야 한다. 이러한 제한점들이 개선되고 가정들이 지켜진다면 본 프레임워크는 현실적으로 웹 서비스 애플리케이션의 동적 성장에 기여할 것으로 기대된다.

본 프레임워크는 웹 서비스 기반 재사용을 가정하고 웹 서비스 애플리케이션의 동적 성장을 지원한다. 즉 서비스 저장소에 서비스들이 존재한다고 가정하고 서비스 발견, 매칭, 재구성을 지원한다. 따라서 본 연구가 일반적인 상황에서 동적 성장을 제공하려면 서비스 저장소에 충분히 다양한 서비스들이 등록되어 있는 것을 요구한다. 또한 유즈케이스 모델로부터 유도되는 서비스 설명은 개발자가 상황을 인식하고 판단하여 가장 적합한 서비스를 선택하지만, 센서 자료로부터 유도된 서비스 설명은 프레임워크 내부에서 상황을 인식하고 판단하여 적합한 서비스를 제공한다. 그러기 위해서는 상황에 대한 정보와 그러한 상황을 추론할 수 있는 지식이 반드시 필요하다. 따라서 일반적인 상황에서 동적 성장을 지원하기 위해서는 폭넓은 상황 정보와 다양한 추론 지식이 필요하다. 본 프레임워크는 서비스 저장소, 상황정보, 추론 지식을 활용하는 토대를 제공하고 있으므로 다양한 서비스, 상황 정보, 추론 지식을 제공한다면 일반적인 상황에서 동적 성장을 지원한다.

본 프레임워크의 장점은 앞에서 기술한 가정이 유효하고 제한점이 개선된다면 웹 서비스 애플리케이션의 반응 적응시 개발자의 개입을 줄이고, 예측 확장시 서비스 기반 재사용을 위한 체계적인 접근 방법을 제시함으로써 시간과 노력을 줄일 수 있다는 것이다. 또한 단점은 현실적으로 인터넷에 분산되어 있는 웹 서비스를 검색하여 바인딩 하는 작업은 시간이 많이 걸린다는 것이다. 에이전트를 사용한 시맨틱 검색 작업 또한 시간이 소요된다. 이러한 요인들로 인해서 프레임워크에서 수행되는 동적 성장 소요 시간을 정확히 예측하기 어렵다. 특히 반응 적응의 경우 시간 요소가 중요한 임베디드 시스템에 적용하는 것은 한계가 있을 수도 있다.

6. 결론 및 향후 연구

본 논문에서는 GTM을 사용하여 요구사항을 기술하며, OWL-S로 웹 서비스를 기술하여 실행시간에 동적으로 서비스들을 찾아 적응하고 기능을 확장하는 성장 프레임워크를 제안하고, 프레임워크의 프로토타입을 구현하였다. 이 프레임워크는 SI, SR와 SD의 세 계층으로 구성된다. SI 계층은 유즈케이스 모델로부터 목표와 태스크들을 유도하여 매핑 규칙을 통해 활동도로 변환하고, 온톨로지 저장소의 참조를 통해 서비스 설명들을 생성한다. 웹 서비스의 특징은 유즈케이스 모델로부터 생성된 GTM과 활동도에 포함된다. 중개 에이전트는 온톨로지의 개념을 사용하여 요청 정보와 서비스를 타입 매칭하고 결과를 정렬한다. SD 계층에서 웹 서비스 검색은 에이전트를 사용함으로써 지능적인 서비스 검색을 가능하게 하였다. SR 계층의 재구성 관리자는 SD 계층으로부터 결과를 받아서 ESB를 이용하여 동적으로 애플리케이션을 재구성한다.

본 프레임워크에서 QoS 정보는 서비스의 신뢰성을 높이는 수단이기 때문에, 웹 서비스에 맞는 정보들이 포함되어야 하고 또한 향상된 필터링을 제공하기 위해 도메인에 특화된 정보들을 제공해야 한다. 그러나 현재 QoS 정보는 이러한 정보를 제공하는데 충분치 않으므로 보완되어야 한다. 또한 서비스 검색이 인터넷을 통해 이뤄지기 때문에 빠른 응답을 요구하는 환경에서는 제약을 갖는다.

향후 연구로는 서비스를 검색할 때 서비스의 매칭이 완전하게 일어나지 않고 부분적으로 매칭되는 서비스 검색에 대한 연구가 진행되어야 할 것이다. 또한 검색된 서비스들을 조합하여 실행할 방법에 대해서도 연구가 이뤄져야 한다. 그리고 신뢰할 수 있고 효과적인 성장을 위해 재사용 가능한 서비스 패턴에 대한 연구가 이뤄져야 할 것이다. 마지막으로 본 프레임워크의 유효성을 보이기 위하여 다양한 시나리오에 기반한 프로토타입을 보이는 것이 필요하다.

참 고 문 헌

[1] M. P. Singh and M. N. Huhns, Service-Oriented Computing, John Wiley & Sons, Ltd., 2005.

[2] R. Groenmo and M. Jaeger, "Model-Driven Semantic Web Service Composition," Proc. of the 12th Asia-Pacific Software Engineering Conference, 2005.

[3] M. Uschold and M. Gruninger, Ontologies: Principles, Methods and Applications, Knowledge Engineering Review, ISI Web of Knowledge, Vol. 11, No.2, pp. 93-136, 1996.

[4] W3C, OWL-S: Semantic Markup for Web Services, Nov. 2004.

[5] D. Fensel and C. Bussler, "The Web Service Modeling Framework," Proc. of International Semantic Web Conference, 2002.

[6] S. Chaiyakul, K. Limapichat, A. Dixit and E. Nantajeewarawat, "A Framework for Semantic Web Service Discovery and Planning," Proc. of IEEE

Conference on Cybernetics and Intelligent Systems, pp. 1-5, June 2006.

[7] ESB: Enterprise Service Bus, http://www-07.ibm.com/software/kr/soa/foundation/esb.html

[8] W. Yu, J. Li, G. Butler, "Refactoring Use Case Models on Episodes," Proc of 19th IEEE International Conference on Automated Software Engineering, 2004.

[9] K. Rui and G. Butler, "Refactoring use case models: the metamodel," Proc. of the 26th Australasian computer science conference, Vol. 16, pp 301-308, 2003.

[10] M. Moon, K. Yeom and H. Chae, "An Approach to Developing Domain Requirements as a Core Asset Based on Commonality and Variability Analysis in a Product Line," IEEE Transactions on Software Engineering, Vol. 31, Iss. 7, pp. 551-569, July, 2005.

[11] 신수미, 류범종, 신동구, "유즈케이스 분석을 통한 효율적인 서비스 모델링 방법," 한국정보과학회 한국컴퓨터종합학술대회 논문집, 제34권, 제1호(B), 2007.

[12] Y. Kim, H. Yun, "An Approach to Modeling Service-Oriented Development Process," IEEE International Conference on Services Computing, pp. 273-276, Sept. 2006.

[13] M. Lin, H. Guo and J. Yin, "Goal Description Language for Semantic Web Service Automatic Composition," Proc. of the Symposium on Applications and the Internet, 2005.

[14] J. Timm and G. Gannod, "A Model-Driven Approach for Specifying Semantic Web Services," Proc. of the IEEE International Conference on Web Services, 2005.

[15] H. Koo and I. Ko, "A repository framework for self-growing robot software," Proc. of Asia-Pacific Software Engineering Conference, 2005.

[16] M. Hinchey and R. Sterritt, "Self-Managing Software," IEEE Computer, Vol. 39, Iss. 2, pp. 107 - 109, Feb. 2006.

[17] 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.

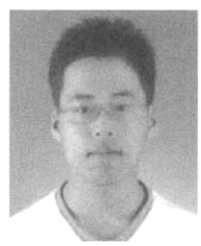
[18] S. Chang, J. Bae, W. Jeon, H. La and S. Kim, "A Practical Framework for Dynamic Composition on Enterprise Service Bus," Proc. of IEEE International Conference on Services Computing, 2007.

[19] 이창호, 김진한, 박동민, 이병정, "지능형 로봇의 적응성을 위한 OWL-S와 에이전트 기반의 확장된 웹 서비스," 한국정보과학회 소프트웨어공학회지, Vol. 19, No. 3, pp.35-44, Sep. 2006.

[20] J. Kim, J. Lee and B. Lee, "Runtime Service Discovery and Reconfiguration using OWL-S based Semantic Web Service," Proc. of IEEE 7th International Conference on Computer and Information Technology, to be published. 2007.

[21] F. Bellifemine, A. Poggi and G. Rimassa, "JADE, A FIPA2000 Compliant Agent Development Environment," AGENTS'01, May 2001.

[22] FIPA: Foundation for Intelligent Physical Agents, http://www.fipa.org/07/2006.



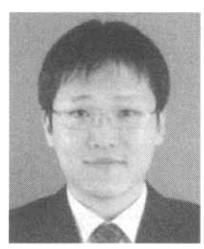
이 창 호

e-mail : leechangho@venus.uos.ac.kr

2006년 서울시립대학교 컴퓨터통계학과 (이학사)

2006년~현재 서울시립대학교 대학원 컴퓨터통계학과(석사과정)

관심분야 : 소프트웨어 진화, 시맨틱 웹 서비스



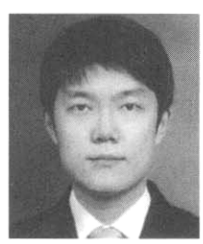
김 진 한

e-mail : kimjinhan@venus.uos.ac.kr

2006년 서울시립대학교 컴퓨터통계학과 (이학사)

2006년~현재 서울시립대학교 대학원 컴퓨터통계학과(석사과정)

관심분야 : 소프트웨어 진화, 시맨틱 웹 서비스



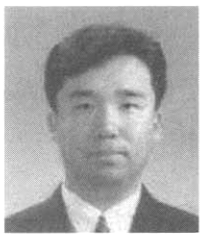
이 재 정

e-mail : jaejeong2@gmail.com

2007년 동국대학교(학사)

2007년~현재 서울시립대학교 대학원 컴퓨터통계학과(석사과정)

관심분야 : 소프트웨어 진화, 시맨틱 웹 서비스



이 병 정

e-mail : bjlee@uos.ac.kr

1990년 서울대학교 계산통계학과(이학사)

1998년 서울대학교 대학원 전산과학과 (이학석사)

2002년 서울대학교 대학원 컴퓨터공학부 (공학박사)

1990년~1998년 ㈜ 현대전자 소프트웨어연구소 연구원

2002년~현재 : 서울시립대학교 컴퓨터과학부 교수

관심분야 : 소프트웨어 진화, 소프트웨어 품질, 소프트웨어 개발 방법론