DOI: 10.3745/KIPSTB.2008.15-B.5.449

비집중화된 조정 기법을 이용한 에이전트 기반 웹 서비스 지원

정 슬 기[†] · 이 태 경[††]

요 약

본 논문은 웹 환경에서 기존의 웹서비스 골격과 기능을 유지하면서, 에이전트 사이를 조정하여 웹 서비스를 지원하는 새로운 시스템을 설계하고 구현하였다. 본 논문에서는 제안한 LDL을 사용함으로 에이전트가 사용자의 요청을 동적으로 수행 할 수 있었고, 그룹화된 에이전트들이 공동의 목표를 상호 협력적으로 수행하기 위해 위임 과정을 수행함으로써 멀티 에이전트에 의한 웹서비스의 가능성을 보였다. 구현된 시스템의 검증을 위해서 수식계산 시나리오를 사용하였으며, 실험결과 제안한 새로운 시스템은 웹 서비스를 위해 필요한 모듈 검색의 시간 비용이 절감됨을 알 수 있었으며, 일반 사용자도 쉽게 접근이 가능하였다.

키워드 : 에이전트시스템, 웹 서비스, 조정

Agent_based Web Service Support using Decentralized Coordination Technique

Seulki Jung[†] · Taekyung Lee[††]

ABSTRACT

This paper proposed the agent-based web service platform that works toward providing coordination technique on web environment. The system described in this paper is designed and implemented to support web services between agents while maintaining web service module. The agent perform user's request dynamically with the proposed LDL, and it offer possibility of coordinated web services by performing commitment process to perform their goals with mutual cooperation for grouped agents. On experiments to check an implemented system, we found that the cost of module retrieval time is reduced and users are accessible.

Keywords : agent system, web service, coordination

1. 서 론

현재의 웹 서비스 환경에서 개발자들이 특정 서비스를 배포 할 경우에는 항상 로직을 새롭게 만들어야 하는 불편을 갖고 있다. 이런 상황이 발생하는 이유 중 하나는 다른 제공자가 배포한 서비스가 개발자의 새로운 서비스에 필요한 경우 직접 찾아 자신의 서비스와 연동시킨 후 자신의 서비스를 배포해야 하기 때문이다. 또 다른 하나는 웹 서비스 중 기본적 서비스 모듈을 이미 개발자가 배포하였다고 하여도 만약 모듈을 사용하는 방법을 모른다면 일반 사용자들은 여전히 사용하기 힘들다는 문제점이 있다.

이런 이유로 본 논문에서는 기존 웹서비스의 문제점들을 해결하기 위한 방법으로 웹서비스를 지원하는 멀티 에이전트시스템을 제안하였다. 이를 위하여 비 집중화된 방식을 사용하여 에이전트간의 정보 교환과 조정(coordination)을 위해 그룹화 정책을 도입하여 해결하고자 하였다.

이 문제를 해결하기 위하여 본 논문에서 사용하는 이론적 배경에 대한 연구로 웹 서비스 부분과 멀티 에이전트시스템을 연구하였다[4]. 기본이 되는 웹 서비스의 구성 요소인 웹 서비스를 분류하고 저장해 검색하는 역할을 하는 UDDI는 W3C에서 정의한 표준 문서인 UDDI V3를 이용했다[12]. 또한 멀티에이전트 시스템에 대한 이론적 연구는 관련 논문들을 살펴보았고 특히 에이전트 사이의 조정[5,6,8,9] 개념에 대하여 조사하였다.

실험에서는 사용할 웹 서비스를 지원하는 멀티 에이전트 시스템을 위해 웹 서비스를 요청해 사용하는 요청자 지원 에이전트를 설계하고 구현하였다. 완벽한 웹서비스를 위해서 개발자를 지원하는 제공자 지원 에이전트도 필요하지만 본 논문에서는 이 에이전트는 다루지 않는 대신에 각 에이전트들이 맡은 업무를 수행할 수 있게 하기 위해 비 집중화된 조정 알고리즘을 채용하여 구현하였다. 웹 서비스 환경에서 특정 도메인에 대해 제한 받지 않기 위해 수식계산에 대한 시나리오를 사용하였다.

2. 이론적 배경 및 접근방법의 타당성

2.1 웹 서비스와 UDDI 관계

웹에서는 직접 브라우저를 통하여 기존의 서비스를 그대

† 정 회 원 : 연세대학교 컴퓨터과학과 박사과정
†† 종신회원 : 동국대학교 컴퓨터멀티미디어학부 교수
논문접수 : 2008년 3월 3일
심사완료 : 2008년 4월 29일

로 사용할 수 있기 때문에 사용자에게 편리한 환경을 제공하고 있다. 웹의 편리성에 따라 표준 프로토콜을 이용한 데이터 교환과 통합을 위한 기술로 등장한 것이 웹 서비스이다[4]. 현재의 웹 서비스는 통신을 위해서 HTTP 표준 프로토콜과 XML을 사용하며, 신규표준으로는 인터넷 상에서 XML 데이터를 교환하기 위해 메세징 프레임웍을 정의하는 SOAP (Simple Object Access Protocol) 개념을 도입하였다[4].

웹 서비스는 서비스 배포자인 개발자와 서비스 사용자와 서비스 정보 저장소인 UDDI(Universal Description, Discovery & Integration)로 구성된다. 개발자와 사용자 및 UDDI는 WAN 환경에 연결되어 있다. 개발자는 개발된 서비스를 배포할 때 서비스 요청자들이 검색을 할 수 있도록 UDDI에 WSDL(Web Service Definition Language)의 주소와 비즈니스, 서비스 정보 등을 등록 한다. 개발자가 개발할 때는 개발 툴을 사용하여 기존에 배포되어 있는 사용 가능한 웹 서비스를 UDDI를 통해 검색하여 재사용 할 수 있다. 서비스 요청자들은 장치나 브라우저를 통해 WAN환경에 있는 어플리케이션에 접근한다. 어플리케이션은 프로그래밍된 로직대로 사용 가능한 웹 서비스들을 호출하여 얻은 결과물들을 취합해 서비스 요청자에게 제공한다.

UDDI는 크게 3가지 기능으로 이루어진다. 첫째 웹 서비스의 개념적 형태를 저장하고 알려줘야 하며, 둘째 프로그래밍 가능한 인터페이스인 API를 제공해야 하며, 셋째 요청과 결과에 대해서는 XML 스키마와 관련된 문서로 이루어져야 한다[12]. UDDI의 접근은 JAVA나 .NET플랫폼을 지원하는 SDK 라이브러리가 공개 되어 있으며 SOAP을 통해 접근한다.

2.2 조정 관련연구

에이전트시스템의 주된 연구는 지능형 인지모델, 시스템 구조, 조정 메카니즘(coordination mechanism)이라는 세 가지 양상을 띤다. 이 중 조정과 관련된 연구들을 살펴보면 E.H. Durfee와 V.R. Lesser는 PGP(Partial Global Planning)를 발표했다[6]. 이 논문에서는 중간 레벨 목표구조의 의사소통에 기반을 두어 목표들을 예측하는 방법을 사용하였으며, PGP의 기초는 에이전트 목표 구조들의 성질을 구축하고 유지하며 정보를 교환하는 작업을 수행한다.

Keith S. Decker와 Victor R. Lesser는 GPGP(Generalizing the Partial Global Planning) 알고리즘을 발표했다[5]. 이 논문에서는 보다 정보를 추상적이고 계층적으로 조직하여 통신하기 위한 접근으로 확장된 PGP 알고리즘을 이용하고, 에이전트 간의 조정 관계를 일반화(generalization) 시켰다. GPGP는 도메인에 독립적인 조정 기법을 지니며, 로컬 스케줄링으로 부터의 조정 프로세스를 분류하는 기능을 갖는다. 이들은 또 다른 논문에서 모든 특정한 조정 메커니즘은 정보수집 행동의 조정, 로컬 스케줄러 요청, 실행을 위한 스케줄 선택, 테스크 그룹의 처리를 종료할 시점의 결정이라는 네 가지 공통된 성질을 바탕으로 한 조정 기법을 사용한 알고리즘을 발표하였다.

2.3 에이전트시스템 접근의 타당성 검토

에이전트시스템으로의 접근에 대한 이론적 타당성의 근거를 찾아보면 다음과 같다. 첫째는 웹 서비스를 위해서 SOAP, OWL등의 기술을 사용하며, 웹 기술 발전을 위해서는 지능형 에이전트의 사용이 궁극적인 목표에 적합할 수 있다. 시멘틱 웹을 연구 중인 Tim Berners-Lee가 2001년 발표한 논문에서 위의 문제점들에 대해서 에이전트가 대안이 될 수 있다고 말하였으며, 시맨틱 웹의 한 부분인 온톨로지만으로는 한계성이 있다고 언급하였다[11].

둘째로는 지능형 웹 서비스를 위해 첫 번째에서 제시한 지능형 에이전트의 개념방법도 있지만 멀티 에이전트시스템의 개념으로 접근하는 것이 더욱 효율적이며 타당하다는 것이다. Peter Stone의 "멀티에이전트시스템"은 단일 에이전트와 멀티 에이전트 사이의 차이와 디자인에 있어서 요구되는 사항, 적용 가능한 솔루션 등의 특징에 대해서 체계적으로 언급하고 분류했다[7]. 이 분류법을 통해 보면 다양한 클라이언트 환경에서의 서비스 접근들을 지원하는 지능형 에이전트들이 있을 때, 이 에이전트들이 독립된 에이전트로 존재 하는 것보다 에이전트간 의사소통 지원이 좋은 멀티 에이전트시스템 기반이 단일 에이전트 기반보다 효율적이다.

추가적으로 웹 서비스 디자인에 대한 지능적 방법론과 관련하여 Terry Payne이 2008년 발표한 논문에서는 에이전트를 이용한 웹 서비스 디자인에 대해 둘 사이의 근본적인 차이점들을 정의했다[10]. Chhabra는 웹 서비스 엔지니어링을 위한 에이전트-지향 방법론을 제안 했다[3].

3. 에이전트 기반의 시스템 설계

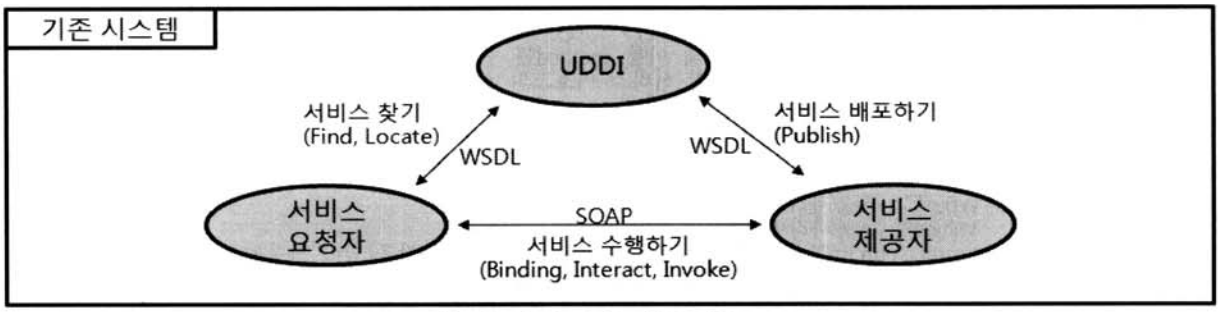
3.1 시스템 설계와 프로세스

3.1.1 설계 원칙과 기법

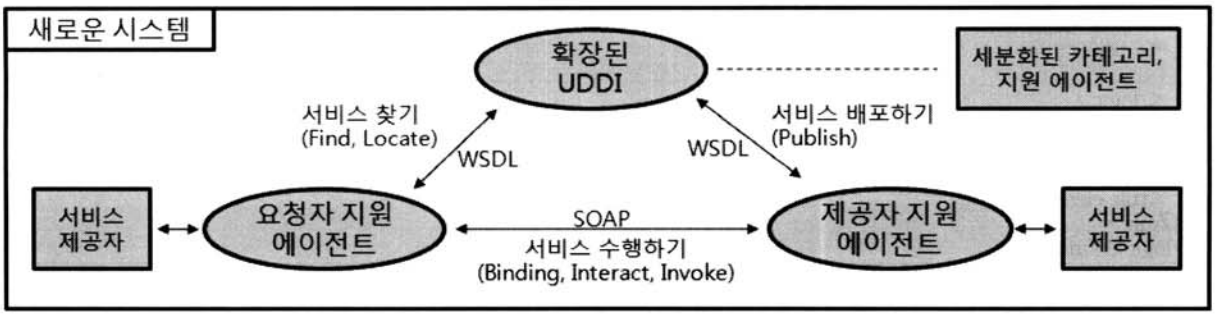
본 논문에서 구성하는 새로운 시스템은 기존과 같이 서비스 요청자(유저)와 서비스 제공자(개발자), UDDI로 구성한다. 그러나 본 논문에서는 새로운 시스템 구축을 위해 추가하는 요소로 요청자 지원 에이전트와 제공자 지원 에이전트가 있으며, 확장된 UDDI는 에이전트를 지원하기 위해 추가적인 카테고리 자료를 저장하도록 한다.

웹 서비스를 위해서 WAN상의 모든 사용자들은 접근 가능해야 함을 원칙으로 한다. 그러므로 본 논문에서 설계된 새로운 시스템도 모든 사용자들은 접근 가능하도록 설계한다. 관련된 에이전트의 정보를 모두 저장하고 변화를 기록하여 에이전트간의 조정을 위해 사용한다.

본 논문에서는 (그림 1b)의 새로운 시스템 중에 우선 요청자 지원 에이전트를 설계하고 구현한다. 새로운 시스템은 (그림 1a)와 같은 기존 시스템의 SOA 개념과 작동 방식을 확장한 형식이다. 형식 유지를 위해서는 기존 UDDI의 API들을 사용하며 SOAP를 이용해 상호간 통신을 하도록 한다. 완벽한 웹서비스를 위해서 개발자를 지원하는 제공자 지원 에이전트도 필요하지만 본 논문에서는 다루지 않는 대신에 각 에이전트들이 맡은 업무를 수행할 수 있게 하기 위해 비



(그림 1a) 기존 시스템의 개념도



(그림 1b) 새로운 시스템의 개념도

집중화된 조정 알고리즘을 구현한다.

3.1.2 시스템 프로세스

기존 시스템과 달리 새로운 시스템에서는 에이전트 스스로 서비스의 검색, 요청, 결과 받기를 처리해야 한다. 이를 위해 (그림 2)와 같은 프로세스 모듈을 사용함으로써 기존 시스템의 목적에 부합하며, 추가적으로 웹 서비스를 이용하는 유저들의 편의를 증진 시키고, 시스템의 효율을 높이고자 한다.

3.2 LDL 설계

멀티 에이전트의 서브골 생성과 수행을 목적으로 하는 새로운 언어가 필요하다. 본 논문에서는 에이전트의 컨트롤러가 새로운 서브골을 비즈니스 로직에 따라 동적으로 생성하고 수행을 위해 사용 가능한 LDL(Logic Definition Language)을 설계하였으며, 이는 서브골에 추가되거나 변경될 로직과 규칙(Rule)을 기술한다. 구성 형식은 (그림 3)과 같다.

(그림 4)는 LDL로 표현한 서브골을 추가한 예이다. subGoalAdd가 LDL의 목적이며 해당 서브골이 수행할 서비스의 메소드는 mul이다. mul 메소드는 math_operation임을 알리고 있다. 이 정보를 이용해 UDDI에서 해당 서비스 메소드를 검색 하여 사용 한다. 이 서비스골의 우선순위는 2이며 작업 번호는 2이다. 서비스골이 수행되는 시점에서 운선순위가 낮은 순서부터 하위골들을 수행 하게 되며, 같은 우선순위인 경우 낮은 작업 번호부터 수행 한다. 예 에서는 서브골에 2가지 파라미터가 필요하며 첫 번째 파라미터는 더블형의 5, 두 번째 파라미터로 3번째 서브골의 수행 결과를 더블형으로 사용 하라는 의미이다.

1. (사용자의 요구 받기) 서비스 요청자가 요청 에이전트에게 업무를 요구
2. (위임 요청 처리) 프로세스1을 확장 하여 요청자의 요구가 아닌 다른 에이전트로 부터의 업무요청(위임)을 수행
3. (업무 로직 요청) 에이전트는 업무를 수행하기 위해 UDDI를 통해 업무 로직을 요청
4. (서브골 생성) 에이전트는 발견된 업무로직에 따라 서브골을 생성
5. (서브골 수행) 에이전트는 서브골을 수행하기 위해 UDDI를 통해 해당 서비스를 찾기
6. (결과 종합) 에이전트는 모든 서브골을 발견된 서비스를 통해 수행하고 결과를 종합해 요청자가 요구한 업무수행을 완료
7. (위임 요청) 프로세스6을 확장해 요청 에이전트는 자신외의 하나 이상의 다른 에이전트들과 상호작용하여 결과 값을 수집
8. (에이전트 그룹화) 프로세스7을 위해 에이전트는 그룹화 수행
9. (지식 저장) 서브골 수행시 발견한 서비스의 정보들을 저장해 향후 UDDI로 질의 횟수를 줄임

(그림 2) 시스템 프로세스

```
<logic:definitions>
  <logic>                              //0개 이상의 <oper>~</oper>로 구성
    <oper name="" method="">           //name은 작업의 이름, method는 수행할 서비스 이름
          desc=""                      //desc는 수행할 서비스 메소드의 간략한 설명
          sequence=""                  //실행 순서인 우선순위
          number=""                    //작업의 인덱스 번호
          parallel="">                 //속성 값이 true면 분할 가능한 작업
      <parm:element number="">         //몇 번째 파라미터 값인지를 표시
                    type="">           //파라미터 타입
                    value="">          //파라미터 값
                    target_source="">  //파라미터 값이 다른 작업의 결과 값을 받아 사용될
                    target_number="">  경우 작성, target_source가 서브골의 결과 값일 경우
                                       에는 'subGoal' ,target_number는 해당 oper의 속성 값인 number
      </parm:element>
      .....
      <parm:element >                  //0개 이상
      </parm:element>
    </oper>
    .....
    </logic>
  <rule>                               //0개 이상
    <rule:element name=""
                  level="">
    <rule:element>
  </rule>
</logic:definitions>
```

(그림 3) LDL 형식

```
<oper name="subGoalAdd" method="mul" desc="math_operation" sequence="2" number="2">
  <parm:element number="3" type="double" value="5">
  </parm:element>
  <param:element number="4" type="double" target_source="subGoal" target_number="2">      </parm:element>
</oper>
```

(그림 4) LDL의 Oper부분 예

3.3 에이전트 조정 방식

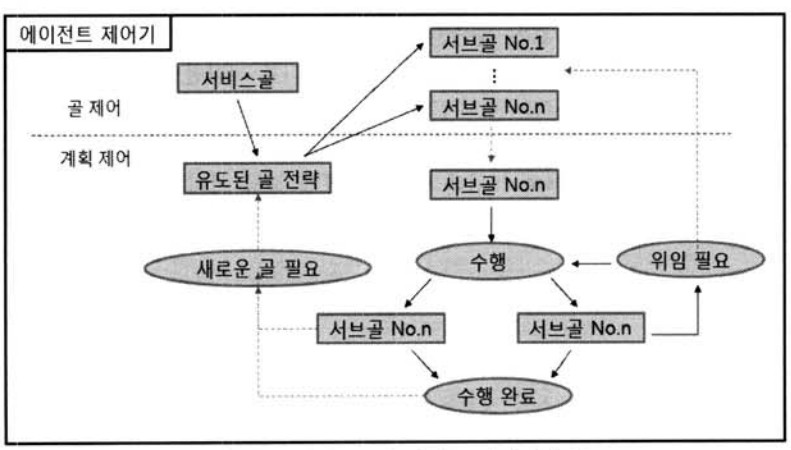
3.3.1 에이전트 정책

본 논문에서는 요청자 조정에이전트, 제공자 조정에이전트를 두어 집중화된 방식을 적용한다면 관리해야 할 에이전트의 개수로 인해 과도한 로드가 발생하므로 비 집중화된 방식을 사용하며, 에이전트 사이의 정보 교환과 조정을 위해 그룹화 정책을 도입한 에이전트시스템을 제안한다.

요청자 에이전트는 사용자의 패턴에 의해 관심 분야를 설정 할 수 있다. 같은 관심 분야를 지닌 관리하는 웹서비스가 필요하다. 이 웹서비스는 그룹에 등록하는 서비스와, 그룹 내의 각 에이전트 IP주소를 제공하는 서비스가 있어야 한다. 그룹별로 그룹 내에 등록된 에이전트 정보 테이블 만을 기록한다. 에이전트들은 자기가 등록된 그룹 내의 다른 에이전트들의 IP정보를 가져와 그룹 리스트에 저장 한다. 그룹 리스트 내의 에이전트들에게는 자신이 휴식 상태인지 아닌지를 주기적으로 알린다.

3.3.2 에이전트 제어기

각 에이전트들이 가지는 제어기의 구조는 (그림 5)에 보였으며, 프로세스1에서 입력된 요구는 골 제어에 의해 서비



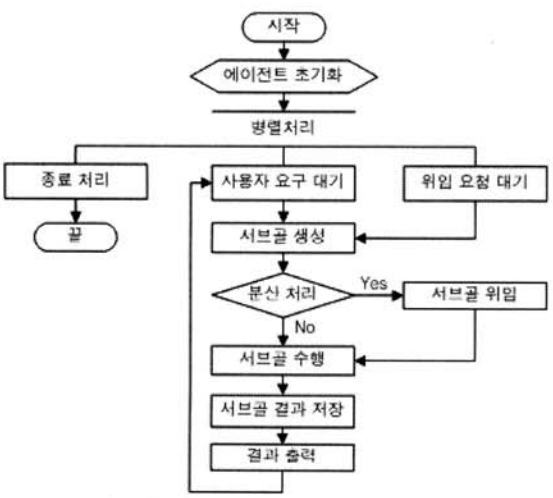
(그림 5) 각 에이전트들이 가지는 제어기의 구조

스골로 등록 된다. 계획제어를 통해 서비스골은 유도된 골 전략으로 세분화 된다. 이때 유도된 골 전략을 얻기 위해 시스템 프로세스3에서 언급한 비즈니스 로직을 UDDI를 통해 찾는다.

유도된 골 전략에 의해 서비스골 수행을 위해 서브골들로 나누어진다. 각 서브골들은 순차적으로 수행 되거나 AND 연산 되어 질수 있고, AND 연산을 효과 적으로 수행하기 위해 다른 에이전트에게 특정 서브골을 위임 할 수 있다. 위임이 필요한 경우 그룹 리스트 상의 휴식상태 에이전트를 찾아 위임요청을 하여 자신의 프로세스를 분할하여 처리할 수 있다. 즉 모든 에이전트는 자기 스스로가 조정 에이전트가 될 수 있다. 본 논문의 실험을 위한 그룹화방법은 프로세스8에서 기술된 방법을 따른다.

3.4 에이전트 프로세스

제안하는 새로운 시스템에서의 에이전트 프로세스는 (그림 6)의 흐름도에 따라 작동한다. 에이전트는 크게 두 가지 작업을 한다. 첫 째는 에이전트의 사용자로부터 요구를 입력 받아 처리하는 일이며, 두 번째는 다른 에이전트로 부터의 위임을 요청받아 대신 처리하는 일이다. 위임은 서브골을 위임을 요청한 에이전트로부터 전달 받아 오는 것이므로 LDL을 전송 받게 된다. 분산 처리의 경우 LDL 표현에 따라 병렬 처리가 가능한 서브골에 해당할 경우 처리하게 된다. 자원 검색은 병렬 처리가 필요한 만큼의 사용자 요구 대기 중인 에이전트를 찾는 것이 목적이다.



(그림 6) 에이전트 프로세스 흐름도

4. 구현 및 실험

4.1 UDDI 사용 테이블

본 논문에서는 UDDI의 구현을 위해 공개소스인 JUUDI를 활용했다. JUDDI는 MySQL을 데이터베이스로 사용 하여 배치했다. (그림 7)은 UDDI에서 사용되는 전체 테이블을 보여준다. (그림 8)은 실험에 사용되는 Tmodel의 내용을 보

(그림 7) UDDI에 사용된 데이터베이스 테이블

TMODEL_KEY	AUTHORIZED_NAME	PUBLISHER_ID	OPERATOR	NAME	OVERVIEW_URL	DELETED	LAST_UPDATE
uuid:327A59	Administrator	[illegible]	UDDI.org	uddi-org:operators	http://www.uddi.org/taxonomies/UDDI_	[illegible]	2007-11-17 23:1
uuid:4064C06	Administrator	[illegible]	UDDI.org	uddi-org:owningBusiness	http://www.uddi.org/taxonomies/UDDI_	[illegible]	2007-11-17 23:1
uuid:4E49A8C	Administrator	[illegible]	UDDI.org	uddi-org:iso-ch:3166-1999	http://www.uddi.org/taxonomies/UDDI_	[illegible]	2007-11-17 23:1
uuid:807A2CE	Administrator	[illegible]	UDDI.org	uddi-org:relationships	http://www.uddi.org/taxonomies/UDDI_	[illegible]	2007-11-17 23:1
uuid:8609CB1	Administrator	[illegible]	UDDI.org	dnb-com:D-U-N-S	http://www.uddi.org/taxonomies/UDDI_	[illegible]	2007-11-17 23:1
uuid:A035A00	Administrator	[illegible]	UDDI.org	uddi-org:general_keywor	http://www.uddi.org/taxonomies/UDDI_	[illegible]	2007-11-17 23:1
uuid:B1B1BAF	Administrator	[illegible]	UDDI.org	thomasregister-com:suppli	http://www.uddi.org/taxonomies/UDDI_	[illegible]	2007-11-17 23:1
uuid:C0B9FE1	Administrator	[illegible]	UDDI.org	ntis-gov:naics:1997	http://www.uddi.org/taxonomies/UDDI_	[illegible]	2007-11-17 23:1
uuid:C1ACF2(	Administrator	[illegible]	UDDI.org	uddi-org:types	http://www.uddi.org/taxonomies/UDDI_	[illegible]	2007-11-17 23:1
uuid:CD1532!	Administrator	[illegible]	UDDI.org	unspsc-org:unspsc	http://www.uddi.org/taxonomies/UDDI_	[illegible]	2007-11-17 23:1
uuid:DB77450	Administrator	[illegible]	UDDI.org	unspsc-org:unspsc:3-1	http://www.uddi.org/taxonomies/UDDI_	[illegible]	2007-11-17 23:1
uuid:E59AE32	Administrator	[illegible]	UDDI.org	uddi-org:isReplacedBy	http://www.uddi.org/taxonomies/UDDI_	[illegible]	2007-11-17 23:1
uuid:math_cal	Administrator	[illegible]	tesk.sk	math_calculation_routine	http://211.186.5.216:8100/test_logic/[illegible]	[illegible]	2007-12-10 12:4
uuid:math_op	Administrator	[illegible]	test.sk	math operation	http://211.186.5.216:8100/WebSite3/[illegible]	[illegible]	2007-12-10 20:2
uuid:passing_	Administrator	[illegible]	tesk.sk	passing_math_expression	http://211.186.5.216:8100/passing_ma	[illegible]	2007-12-10 14:2
uuid:test	Administrator	[illegible]	test.sk	find business logic	http://211.186.5.216:8100/test3/test3	[illegible]	2007-11-27 01:2

(그림 8) Tmodel 테이블

여준다. OVERVIEW_URL에는 해당 서비스의 WSDL 주소 값이 들어있다.

4.2 요청자 지원 에이전트 프로세스 과정

4.2.1 원하는 작업의 서비스 찾기

에이전트에서 UDDI의 정보를 검색하기 위해서는 Microsoft UDDI SDK를 활용하며, 이를 통해 웹 서비스의 기술 문서인 WSDL의 위치를 저장하고 있는 Tmodel을 찾는다(Find). Tmodel 테이블의 서비스 이름을 사용자질의에 대한 의미분석을 통해 검색된 해당 주소의 WSDL파일을 임시 텍스트 파일로 저장을 한다(Locate).

여기서 얻어진 WSDL 파일을 파싱해서 사용할 웹 서비스 접근주소 및 사용할 메소드와 사용 되어질 파라미터 정보를 얻어야한다. (그림 9)는 파싱에 사용되는 PWSDL 클래스를

```
public class PWSDL{
//PWSDL에서 사용할 WSDL파일을 링크
public PWSDL(FileStream strFileName)
//WSDL에서 사용한 파일 링크를 해제
public void close()
//WSDL에 있는 네임스페이스를 반환
public string GetNamespace()
//WSDL에 있는 서비스 네임을 반환
public ArrayList GetServiceName()
//해당 서비스 네임의 엔드포인트 주소인 서비스주소를 반환
public string GetServiceAddress(string strServiceName)
//WSDL에 있는 서비스에 해당하는 오퍼레이션 네임을 반환
public ArrayList GetOperationName()
//WSDL에 있는 해당 오퍼레이션 네임의 솝 액션을 반환
public String GetSoapAction( String strOperName)
//하나의 오퍼레이션은 인풋과 아웃풋 메시지로 구분 되므로 메시지 네임을 반환
public ArrayList GetMessageName(string strOperName)
//해당 메시지 네임의 파트 네임을 반환
public string GetPartName(string strMessageName)
//해당 파트 네임의 파라미터를 반환
public ArrayList GetParameter(string strPartName)
}
```

(그림 9) PWSDL 클래스

보여준다. PWSDL의 메소드를 사용해 얻어진 WSDL을 파싱하고 얻어진 WSDL의 정보는 인스턴스로 생성되며, 인스턴스는 해당 서비스를 SOAP을 통해 호출할 때 사용된다.

찾아진 WSDL이 수행 되어야 할 서브골는서 사용 가능한 서비스를 가리키고 있는지를 찾아보기 위해 WSDL과 서브골에서 필요로 하는 메소드를 매칭시켜 본다. 찾아진 웹 서비스의 메소드가 있을 때는 저장된다. known_logic.Add()를 통해 찾아진 웹서비스 메소드가 있을 때마다 저장된다. 이는 필요한 메소드를 찾기 위해 UDDI에게 질의한 후 WSDL을 받아와 분석한 후에 알고 있는 메소드에서 먼저 필요한 웹 서비스의 메소드를 검색한 후 탐색이 이루어짐으로 실행시간을 줄일 수 있다.

4.2.2 웹 서비스의 메소드 수행하기

웹 서비스를 수행하기 위해서는 Micosoft SOAP Toolkit을 이용해 SOAP 메시지를 전달한다. SOAP은 HTTP프로토콜을 사용하며, 커넥터는 HTTP커넥터를 이용한다(Interact). 매칭된 웹 서비스의 메소드 이름을 이용해 서비스를 호출하게 되며 서브골에서 필요로 한 파라미터 타입에 따라 단순 값, 인풋 박스, 서브골 수행결과로 파라미터를 사용한다(Invoke). 수행된 결과가 단순 값이라면 결과를 서브골의 결과 값으로 저장을 하고, 서브골의 수행 결과가 새로운 서브골을 생성하도록 유도하는 LDL이라면 서브골을 새로 생성하는 로직을 수행한다(Binding).

4.2.3 서브골의 위임

서브골의 위임은 서브골의 parallel 속성이 참 이고 다른 에이전트의 주소가 등록되어져 있으며 다른 에이전트의 상태가 쉬고 있는 상태일 때만 수행 되어야한다. 실험을 위해서 다른 에이전트의 주소와 포트는 하나만 등록 되어 있으며 다른 에이전트는 항상 쉬고 있는 상태로 설정했다. TCP/IP 소켓통신으로 통신이 이루어지며 LDL형식으로 위임될 서브골을 새롭게 만든 뒤 전달한다. 결과로 전송되어 온 값은 위임되었던 서브골의 결과 값으로 저장된다. 소켓통신을 위해서는 System.Net과 System.Net.Sockets 라이브러리를 사용하였다.

4.2.4 다른 에이전트의 위임 요청 수행

다른 에이전트의 위임 요청을 수락하기 위해서 쓰레드 방식으로 클라이언트의 접속을 대기한다. 클라이언트를 바인딩 했을 때 승인된 클라이언트의 IP를 저장하고 전송받은 LDL을 파싱하여 다른 에이전트 돕기 서브골의 하위 서브골로 생성 한다. 서브골 수행 방법은 위의 (1)과 (2)와 동일한 방법으로 수행된다.

4.3 실험 결과

4.3.1 수식계산 시나리오 적용

컴퓨터가 다수의 수식계산을 처리하기 위한 방법으로는 트리구조를 많이 사용한다. (그림 4)와 같이 서비스골을 다수의 서브골들로 유도하여 처리하는 방법은 복잡한 수식을 하나의 연산자 단위별로 구조화 하여 표현하는 방법과 유사하다고 볼 수 있다. 그러므로 설계된 시스템이 일반성을 가지고 있음을 입증하기 위한 수단으로 수식계산 시나리오를 적용했다.

수식 계산 시나리오는 (그림 2)를 수행하기 위해 (그림 6)의 단계에 맞게 처리 되어야 한다. 우선적으로 (그림 6)의 사용자 요구 입력 단계에 맞게 입력창에 '수식계산'을 입력한다. 의미 파악 단계에서는 입력된 요청에 대해 미리 정의되어 있는 의미파악 루틴을 통해 수식 계산 하기를 얻게 된다. 에이전트는 의미 파악이 제대로 이루어 졌는지 확인하기 위해 사용자에게 질의를 한다. 맞는 결과라면 업무 로직 검색 단계로 가서 UDDI를 통해 수식 계산 하기 와 관련된 로직을 찾는다. 이때 발견된 로직이 있다면 서브골 생성 단계로 넘어가서 업무 로직을 LDL형식으로 받아 온다. 받아온 LDL은 또다른 서브골의 생성을 지시하게 된다. 추가적으로 생성된 서브골들은 수식 입력 받기, 수식 파싱 하기와 수식 처리 하기이다.

서브골 수행 단계에서는 첫 번째 서브골에 따라 수식을 입력 받기 위해 지시사항 설명줄을 'input math expression'으로 변경한다. 사용자가 지시 사항에 따라 인풋박스에 수식을 입력하면 입력된 수식은 두 번째 서브골인 수식 파싱하기의 파라미터가 되어 해당 웹 서비스를 호출하게 된다. 파싱 웹 서비스를 통한 결과는 새로운 서브골들을 만들기 위해 LDL로 전송되고, 이를 이용해 새로운 서브골들이 생성되어지며 생성된 서브골에 따라 반복적으로 수식들이 처리된다.

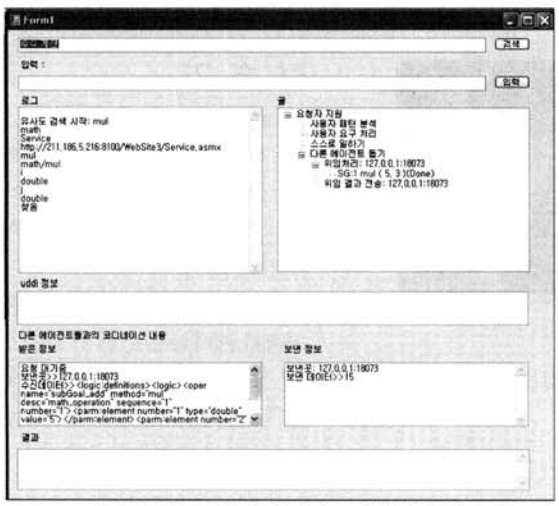
분산 처리 단계에서는 병렬로 처리 가능한(위임이 가능한)서브골이 있을때 에이전트 그룹안의 위임 요청 대기 단계에 있는 다른 에이전트를 찾아 서브골 위임 단계에 따라 위임을 요청하게 된다. 결과 출력 단계에서 처리 완료된 결과 값을 결과 창에 나타 낸다.

4.3.2 요청자 지원 에이전트 결과

요청자 지원 에이전트를 통해 수식계산 서비스를 실행한 결과는 (그림 10)과 같다. 실행 초기의 경우 골에는 시작 시 초기화된 사용자 패턴분석, 사용자요구처리, 스스로 일하기, 다른 에이전트 돕기가 생성된다. 다른 에이전트를 돕기 위해서 다른 에이전트들과의 조정 내용 중 받은 정보에는 요청 대기 중으로 표시된다.

검색창에는 수식계산을 위해 "수식계산"을 입력했다. 수식계산의 의미파악을 통해 요청된 로직을 UDDI를 통해 검색해 서비스를 받아와 수식계산에 따른 서브골은 2개가 추가 생성되었다. 첫 번째 골인 SG:1은 입력 파라미터인 수식을 받기 위해 입력창의 설명을 '입력 : input_math_expression'으로 설정하였고, 수행이 완료되어 서브골에는 (Done)으로 표시되어 있다. SG:2는 인풋 박스의 정보를 passing_math_expression인 웹 서비스를 호출해서 실행하기 위해 대기를 하는 것이며, 입력버튼을 누르면 서브골 SG:2는 수행된다.

(그림 10) 요청자 지원 에이전트의 수식계산이 완료된 결과



(그림 11) 요청자 에이전트 위임의 처리

입력된 수식을 SG:2의 파싱을 통해 얻어진 로직정보를 통해 생성된 새로운 서브골 SG:1~SG:7이 있으며, SG4는 병렬처리가 가능하기에 분할된 트리로 보여 진다. 모든 서브골은 처리가 완료되면 각기 Done으로 표시되며 서비스골의 결과 값으로는 "Result : 16.75" 이다. 로그 정보 중에 있는 KL:0~KL:5은 요청자 지원 에이전트가 가지고 있으며 이미 알고 있는 로직이다. 이것은 (그림 2)의 지식 저장 프로세스로 처리한 결과이다.

4.3.3 요청자 지원 에이전트 위임 처리 결과

다른 에이전트가 위임을 요청 하였을 경우의의 결과는 (그림 11)과 같다. 이는 LDL형태로 보내지며 수락되었을 때 다른 에이전트 돕기의 서브골에 위임처리가 추가된다. 요청된 위임은 서브골로 SG:1에 추가됐다. 처리가 완료 되면 요청된 에이전트에게 결과 값을 돌려준다.

4.4 실험 결과 분석

시스템 프로세스에 따른 수식계산에 대하여 기존 시스템과 제안한 새로운 시스템의 비교는 <표 1>과 같다.

〈표 1〉 기존 시스템과 제안한 새로운 시스템의 비교

	기존 시스템	제안한 새로운 시스템
UDDI 검색 횟수	UDn	UDn + LQn
웹 메소드 접근 횟수	n	n + LQn
사용자의 실행 과정 횟수	UDDI 검색 횟수 + 웹 메소드 접근 횟수	2회(필요한 서비스 요청 1회+ 수식 입력 1회)
총 실행 시간	(n * MRT) +(RD * n)	LMD + (n* MRT)
사용자의 능력	사용자의 높은 수준의 접근 필요	사용자의 쉬운 사용 접근 가능

n = 모든 연산 기호 횟수, UDn = 수식 중 비 중복된 연산 기호 횟수, LQn = 로직 질의 횟수,
MRT = 메소드 실행 시간, RD = 처리 결과 지연 시간, LMD = 로직 생성 지연 시간

기존 시스템의 (그림 1a)의 방법대로 n개의 연산 기호가 들어 있는 수식계산을 웹 서비스를 통할 경우 사용자는 사용된 연산 기호 처리 메소드를 UDDI에서 모두 검색하여야 하며, 검색된 메소드를 이용한 수식계산을 위해서는 첫 번째 메소드 처리할 때부터 n번째 메소드까지 순차적으로 처리해야 하므로 지연시간은 n+(처리 결과 연결 지연시간 * n)이 된다. 제안한 시스템에서는 내부적으로 처리하는 횟수인 UDDI검색 횟수와 웹 메소드 접근 횟수가 LQn만큼 증가한다. 하지만 LQn은 수식 계산 시나리오에서 1회로 처리되며, 복잡한 서비스를 요청할 경우도 총 메소드 수를 UDDI에 질의하는 횟수에 비해 매우 적으므로 무시할 수 있다. 사용자의 실행 횟수는 서비스 요청 1회와 수식 입력 1회인 단 2회 만으로 완료된다. 총 실행 시간의 경우 분산 처리를 할 수 없는 최악의 경우, n번의 메소드 실행 시간 + 로직 생성 지연시간이 걸린다.

제안 시스템은 기존 시스템에 비해 UDDI검색 횟수와 웹 메소드 접근 횟수는 무시 가능한 횟수 만큼만 증가하면서도 요청한 서비스 로직을 실시간으로 스케줄링 해 처리함으로서 실행 시간을 크게 단축 가능하며, 분산 처리를 고려한다면 더욱 빠른 실행 시간을 보장할 것이다. 또한 사용자는 UDDI와 웹 메소드에 직접적으로 접근하지 않아도 되므로 일반 사용자도 사용 가능한 장점이 있다.

5. 결론 및 향후 과제

본 논문에서는 첫째, 멀티 에이전트시스템이 웹서비스 설계에 적합하다는 것을 구현을 통하여 보였으며, 이를 통하여 UDDI의 검색 횟수를 줄였다. 둘째, 에이전트가 필요한 웹 서비스의 요청을 돕기 위해 UDDI의 카테고리 항목을 확장했다. 셋째, 에이전트 사이의 조정을 위한 방법으로 서브골의 위임을 사용함을 보였다. 넷째, 사용자의 요청을 동적인 스케줄 로직으로 만들고 처리하기 위해 LDL을 제안하여 사용하였다.

구현된 시스템의 검증을 위해서 수식계산 시나리오를 사

용하였다. 부분적 수식은 위임을 통해 다른 에이전트와 협력적으로 결과를 얻어 냈으며, 전체 스케줄은 LDL을 통해 동적으로 변경과 수행이 가능했다. 또한 그룹화된 에이전트들이 공동의 목표를 상호 협력적으로 수행하기 위해 위임 과정을 수행함으로써 가능성을 보였다.

실험결과 기존의 웹 서비스 시스템에서는 사용자가 필요한 서비스를 위해 개발자들이 배포해 놓은 모듈들을 직접 검색한 후 메소드들을 호출해야 하므로 시간적 비용이 증가하고 일반 사용자의 접근이 불가능 하였다. 반면에 제안한 새로운 시스템은 일반 사용자의 접근이 가능하고 분산처리와 자체적 로직 생성과 필요한 모듈 검색을 통한 시간적 비용이 절감됨을 볼 수 있었다.

향후 과제로는 제공자 지원 에이전트를 추가하여 구현한다면 보다 완벽한 시스템이 될 것이다. 또한 정교하고 고급화한 그룹화정책을 개발하여 구현한다면 에이전트들이 상호 협력적으로 업무를 더 잘 수행할 수 있을 것이다.

참 고 문 헌

[1] 정슬기, 이태경, "상태 인식에 따른 자율 주행 에이전트시스템", 제25회 한국정보처리학회 춘계학술발표대회 논문집 제13권 제1호, pp.295-298, 2006년 5월

[2] 정슬기, 이태경, "지능형 조정 기법을 이용한 웹 서비스 지원 멀티 에이전트시스템", 2007년 한국멀티미디어학회 추계학술발표대회 논문집 제10권2호, pp.226-229, 2007년 11월

[3] Chhabra, M. Hongen Lu, "Towards Agent Based Web Service," Computer and Information Science, ICIS 2007. 6th IEEE/ACIS International Conference, pp.93-99, July 2007.

[4] Eric newcomer and Greg lomow, "Understandding Soa with web Services," Addison Wesley, April 2005.

[5] Decker, K. and Lesser, V. "Generalizing the Partial Global Planning Algorithm," International Journal on Intelligent Cooperative Information Systems, Vol.1, No.2, pp.319-346. June 1992.

[6] Durfee, E.H. and Lesser, V.R. "Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation," IEEE Transactions on Systems, Man, and Cybernetics, Vol.21, No.5, pp.1167-1183, September 1991.

[7] Peter Stone and Manuela Veloso, "Multiagent Systems: A Survey from a Machine Learning Perspective," Autonomous Robots, Vol.8, No.3 , pp.345-383, June 2000.

[8] Pollock J L, "The logical foundations of goal-regression planning in autonomous agents," Artificial Intelligence, 106: 267-334, 1998.

[9] Shehory O, Kraus S, Yadgar O, "Emergent cooperative goal-satisfaction in large-scale automated- agent systems," Artificial Intelligence, d IO: 1-55, 1999.

[10] Terry R. Payne, "Web Services from an Agent Perspective," Intelligent Systems, IEEE Vol.23, Issue 2, pp.12-14, March-April 2008.

[11] Tim Berners-Lee and James Hendler and Ora Lassila, "The Semantic Web," Scientific American, May 2001.

[12] "Universal Description, Discovery and Integration v3.0.2 (UDDI)". UDDI Spec Technical Committee Draft, Dated 20041019.

정 슬 기

e-mail : lovehard@csai.yonsei.ac.kr

2006년 동국대학교 컴퓨터 멀티미디어학과 졸업(학사)

2008년 동국대학교 대학원 전자계산학과 (공학석사)

2008년~현 재 연세대학교 대학원 컴퓨터과학과 박사과정

관심분야 : 멀티에이전트, 시멘틱 웹 서비스, 모바일 로봇

이 태 경

e-mail : tklee@mail.dongguk.ac.kr

1980년 동국대학교 전자계산학과 졸업(학사)

1985년 숭실대학교 대학원 전자계산학과 (공학석사)

1994년 숭실대학교 대학원 전자계산학과 (공학박사)

1987년~현 재 동국대학교 과학기술대학 컴퓨터멀티미디어학부 교수

관심분야 : 인공지능, 지식공학, 에이전트 시스템