한국지능정보시스템학회논문지 제10권 제2호 2004년 11월(pp.111~131)

OntCIA: 시맨틱 웹 기술 기반의 소프트웨어 변경 영향분석 시스템

송희석
한남대학교 경영정보학과
(hssong@hannam.ac.kr)

소프트웨어 유지보수 단계에서는 고객니즈, 마케팅 정책, 법, 제도의 변화 등으로 인한 다양한 시스템 변경 요구를 수용하여야 한다. 그러나, 소프트웨어의 비가시성문제로 인해 새로운 변경 요구사항 발생 시 수정 대상 모듈을 발견하는데 지대한 시간이 요구될 뿐 아니라 모듈의 재 사용을 어렵게 만들어 중복 모듈이 양산 됨으로써 향후 장애의 근원이 되는 악순환이 전개 된다. 이에 본 연구에서는 시맨틱 웹(Semantic Web) 기술을 활용하여 이동통신사의 과금/청구 도메인의 관리자와 개발자들이 공유하고 있는 개념과 개념간 관계를 명시적으로 표현하고 이를 이용하여 변경대상 모듈을 쉽게 발견 할 뿐 아니라, 발견된 모듈에 대해 구조적 호출 및 조립 관계를 분석하도록 지원하는 온톨로지 기반 변경 영향 분석 시스템(OntCIA; Ontology based Change Impact Analysis System)을 제시한다. OntCIA는 스트링 매칭과는 근본적으로 다른 의미적 모듈검색을 지원하며 잦은 변경이 요구되는 호출 및 조립 구조 정보는 데이터 베이스에서 관리하고 도메인 지식은 온톨로지로 관리함으로써 유지 보수가 용이한 구조를 가진다.

논문접수일 : 2004년 7월 게재확정일 : 2004년 11월 교신저자 : 송희석

1. 서론

소프트웨어 유지보수 단계에서는 고객니즈, 마케팅 정책, 법, 제도의 변화 등으로 인한 다양한 시스템 변경 요구를 수용하여야 한다. 특히, 시스템이 방대하고 기능간 상호의존도가 높은 소프트웨어일수록 특정 소프트웨어 구성요소의 변경이 다른 구성요소에 미치는 영향을 파악하기가 어렵기 때문에 정확한 변경 영향분석은 소프트웨어 유지보수 품질을 확보하는데 매우 중요한 요소가 되고 있다[Law and Rothermel, 2003; Ryder and Tip, 2001; Zhao et al., 2002; Arnold and Bohner, 1993; Bohner and Arnold, 1996]. 대규모 소프트웨어의 변경관리에서 접하는 문제는 빌딩이나, 자동차와는 달리 소프트웨어의 구성이 숨겨져 있다는 사실이다. 개발 당시에 엄격한 개발 문서(분석서 및 설계서)관리를 한다 하더라도 잦은 변경으로 인해 개발 문서가 소스코드와 불일치 함으로 인해 개발 문서를 활용한 소프트웨어 구성의 파악에는 한계가 있다. Devanbu(1991)와 Brooks (1987)도 이러한 문제를 비가시성(Invisibility)의 문제로 지적하고 있다. 소프트웨어의 비가시성문제로 인해 새로운 변경 요구사항 발생 시 수정 대상 모듈을 발견하는데 지대한 시간이 요구될 뿐

아니라 모듈의 재 사용을 어렵게 만들어 중복 모듈이 양산 됨으로써 향후 장애의 근원이 되는 악순환이 전개 된다.

그간 소프트웨어 변경 영향분석을 지원하는 방법은 크게 두 가지 접근법으로 요약된다. 첫 번째 접근법은 설계서 등의 산출물을 소스 코드와 동기화하여 관리함으로써 설계서를 통해 변경에 대한 영향분석을 실시하는 방법이다[Kellens, 2003]. 이는 변경에 따른 소스코드의 수정작업과 동시에 설계서를 일치화 해야 하며 이 과정에서 개발자의 부담을 최소화 할 수 있도록 자동화된 재구조화(Restructuring) 방법을 도입 한다. 그러나 이 또한 코드 수정 외의 추가 부담을 경감시켜줄 수는 있지만 소스코드와 설계서의 완벽한 동기화를 보장할 수 없기 때문에 높은 신뢰도가 유지되는 대규모 소프트웨어의 변경관리에 적용되기에는 한계가 있다. 두 번째 접근법은 소스코드 수준에서 구성요소간 변경 영향을 직접 분석하도록 지원하는 방법이다[Law and Rothermel 2003; Zhao et al., 2002; Ryder and Tip, 2001; Shirabad, 2001]. 이는 자동화된 코드 스캐닝과정을 통하여 모듈간 호출구조와 조립구조를 자동으로 완성함으로써 특정 구성요소의 변경 시 동시 수정이 요구되는 구성요소를 쉽게 판단할 수 있도록 지원한다. 이 접근법은 소프트웨어 구조적 측면의 변경 영향분석에 있어서는 다소 신뢰성이 보장된다. 그러나 이 방법은 호출 및 조립관계를 질의해야 할 대상 프로그램 파일명이 무엇인지 정확히 알고 있어야 사용 가능하다는 단점이 있다. 예를 들면 "신규고객을 등록하는 모듈"을 찾아서 수정해야 하는 경우, 신규고객 등록을 위한 모듈이 "usr/pp_bill/aram_DpRtn" 인지 혹은 "usr/pp_bill/aram_DRtn" 인지 또는 다른 제 3의 모듈인지를 찾을 수 있어야 이 모듈과 관련 호출관계 있는 모듈들을 수정할 수 있다. 변경대상 모듈을 발견하는 것은 여러 개발자가 참여하는 대규모 소프트웨어 일수록 더욱 어려워진다. 이에 본 연구에서는 시맨틱 웹(Semantic Web) 기술을 활용하여 과금/청구 도메인의 관리자와 개발자들이 공유하고 있는 개념과 개념간 관계를 명시적으로 표현하고 이를 이용하여 변경대상 모듈을 쉽게 발견 할 뿐 아니라, 발견된 모듈에 대해 구조적 호출 및 조립 관계를 분석하도록 지원하는 온톨로지 기반 변경 영향 분석 시스템(OntCIA; Ontology based Change Impact Analysis System)을 제시하고자 한다. 본 연구에서 과금/청구도메인은 이동통신사의 과금/청구 프로세스를 대상으로 하며 통화 기록 수집에서 고객정보관리, 사용요금계산 및 청구서 발행 프로세스를 포함한다. 이동통신사의 과금/청구 도메인은 기능간 상호의존도가 매우 높은 복잡한 시스템으로써 소프트웨어 변경 영향분석이 안정적인 시스템 운영에 매우 중요한 분야로 알려져 있다. 본 연구에서 제시하는 시스템은 과금/청구 도메인 지식과 소스 코드의 구조적인 정보를 결합한 검색이 가능하다. 예를 들면, "EBPP(Electronic Bill Presentment and Payment; 전자고지 및 결재)용 파일 생성 모듈이 호출하는 모듈들을 모두 찾아라" 와 같이 도메인 지식과 결합된 소프트웨어 구조정보의 검색이 가능하다. 또한 본 시스템은 질의(Query)에 사용된 문자열(String)에 의한 단순 매칭 검색과는 근본적으로 다른 의미적 검색을 지원한다. 이는 도메인 지식에 대한 기술과 더불어 추론 기능을 활용함으로써 이루어 진다. 예를 들면 관리자가 "복합상품 가입처리 모듈을 검색하라" 라는 질의에 대해 복합상품이 결합상품과 같은 용어이고, 가입처리는 "재가입"과 "신규 가입"으로 나누어 질 수 있다는 것을 기계가 이해하여

"결합상품 재가입 처리 모듈", "결합상품 신규가입 모듈", "복합상품 재가입 처리 모듈", "복합상품 신규가입 모듈" 까지를 검색해 주는 것을 말한다. 결국, 본 연구는 "RDF(Resource Description Framework)와 온톨로지 등의 시맨틱 웹 기술이 대규모 소프트웨어의 변경영향분석에 이슈가 되는 복잡성과 비가시성 문제를 어떻게 해결할 수 있는가?"에 대한 해답을 제시한다.

본 논문은 다음과 같이 구성된다. 2장에서는 시맨틱 웹과 온톨로지 기술에 대해 간략히 소개하고 소프트웨어 변경 영향분석 관련 연구를 소개한다. 3장에서는 소프트웨어 변경 영향분석 시스템의 요구사항을 변경업무 프로세스 조사를 통해 추출하고 있으며, 4장에서는 온톨로지 기반의 변경 영향분석시스템의 구조를 제시하고 기능별 주요 설계 이슈를 소개한다. 마지막으로 5장에서는 본 연구의 결론과 향후 연구과제를 제시한다.

2. 기존 연구

2.1 시맨틱 웹(Semantic Web)과 온톨로지(Ontology)

본 장에서는 본 연구에서 채택하고 있는 시맨틱 웹 기술에 대해 간략히 소개한다. XML (eXtensible Markup Language)이 문서의 구조와 내용을 정의하는 기술요소라면, RDF(Resource Description Framework)[1]는 특정 자원(혹은 문서)에 대한 의미를 주어(Subject), 술어(Predicate), 목적어(Object)에 해당하는 세 가지 요소로 표현한다[Berners-Lee et al., 2001; Davies et al., 2003; Daconta et al., 2003; 최중민, 2003; 조성정과 김진형, 2003; 전종홍 외, 2003]. 여기서 주어와 목적어는 특정 개념을 지칭하고 술어는 개념들 간의 관계를 정의하는데 사용된다. 예를 들면, "고객이 상품을 구매한다" 라는 문장에서 "고객"은 주어, "상품"은 목적어 "구매한다"는 술어로 구분하여 기술함으로써 "상품이 고객을 구매"하는 것이 아니라, "고객이 상품을 구매한다"라는 의미를 기술하게 된다. RDF로 자원을 기술할 때, 일반화된 개념(Generalized Concept; Super Class)과 구체화된 개념 (Specialized Concept; Sub Class)을 정의하고, 개념간 관계에 대한 특성과 제약사항을 정의한 후 이에 근거하여 기술하는 것이 효과적인데 이를 위해 온톨로지(Ontology)가 사용된다. 온톨로지는 여러 가지로 정의되지만 "공유된 개념화(Shared Conceptualization)에 대한 정형화 되고 명시적인 명세 (Formal and Explicit Specification)"라는 Gruber (1993)의 정의가 시맨틱 웹 분야에서 많이 사용된다. 온톨로지에는 계층분류(Taxonomy)와 추론규칙(Inference Rule)에 대한 정의가 포함된다. 계층분류는 개념의 일반화와 구체화를 통한 클래스와 서브 클래스의 정의와 그 들 간의 관계를 정의한다. 추론 규칙은 프로그램이 새로운 사실을 자동으로 추출하거나 제약조건에 맞지 않는 오류를 찾아내는데 이용된다. 온톨로지를 표현하기 위한 언어로는 OWL, DAML+OIL, Ontolingua 등이 있으나, OWL(Web Ontology Language)은 RDF와 RDF스키마 표준에 기반을 두고 이를 확장한 언어로써 W3C에서 가장 최근에 시맨틱 웹을 위한 표준 온톨로지 언어로 권고 된 언어이다. 본 연구에서는 OWL[2]을 이용하여 온톨로지를 표

1) http://www.w3c.org/TR/rdf-primer/

2) http://www.w3c.org/TR/owl-guide/

현하고 RDF로 소프트웨어 자원에 대한 의미를 기술하는 방식을 사용함으로써 과금/청구 도메인의 온톨로지 기반 변경 영향분석 시스템을 제안하고자 한다.

시맨틱 웹은 카탈로그 관리, 데이터 통합, 지식 경영, 메타 데이터 기술, 의미적 검색 등 다양한 응용 분야에서 적용되고 있으며[Reynolds et al., 2001] 최근에는 지리적으로 분산되고 이질적인 정보를 통합하는 실 세계의 응용 사례가 개발되고 있어서 시맨틱 웹의 유용성을 한 차원 높이고 있다[Klein and Visser, 2003; Michalowski et al., 2004; Cost et al., 2002]. 그러나 시맨틱 웹 기술을 소프트웨어 변경 영향분석에 응용한 사례는 아직 보고 되지 않고 있다.

2.2 소프트웨어 변경 영향분석 관련연구

소프트웨어에 대한 잦은 변경은 작게는 의도되지 않은 실행 결과로부터 크게는 치명적인 장애로 이어지기 때문에 변경에 대한 영향분석 연구는 소프트웨어 공학 및 인공지능 연구분야에서 줄 곧 중요한 이슈가 되어 왔다. 먼저 소프트웨어 공학분야에서는 소프트웨어의 구조적인 측면에서 특정 프로시져의 변경이 다른 프로시져에 주는 영향 연구가 90년 이후 활발히 진행되어 왔다[Law and Rothermel, 2003; Ryder and Tip, 2001; Zhao et al., 2002; Arnold and Bohner, 1993; Bohner and Arnold, 1996]. 이를 위해 프로시져간 호출관계를 이용하여 특정 프로시져의 변경이 미치는 영향 범위를 분석하는 호출 그래프(Call Graph) 기반의 영향 분석 기법과 작은 단위로 프로그램을 분할하여 데이터 및 제어의 의존성 관계를 계산함으로써 보다 신뢰성이 높은 분야에서 사용 가능 한 프로그램 분할(Program Slicing) 기반의 영향분석 기법 등이 소개 되었다. 그러나 이러한 연구는 구조적인 측면에서의 변경 영향범위를 파악하는데 주안점을 두고 있기 때문에 비즈니스 도메인에서 사용되는 공유된 온톨로지에 기반한 의미적 검색과 구조적인 측면의 호출 및 조립 관계를 결합하여 변경에 대한 영향분석을 하는 본 연구와는 차이가 있다.

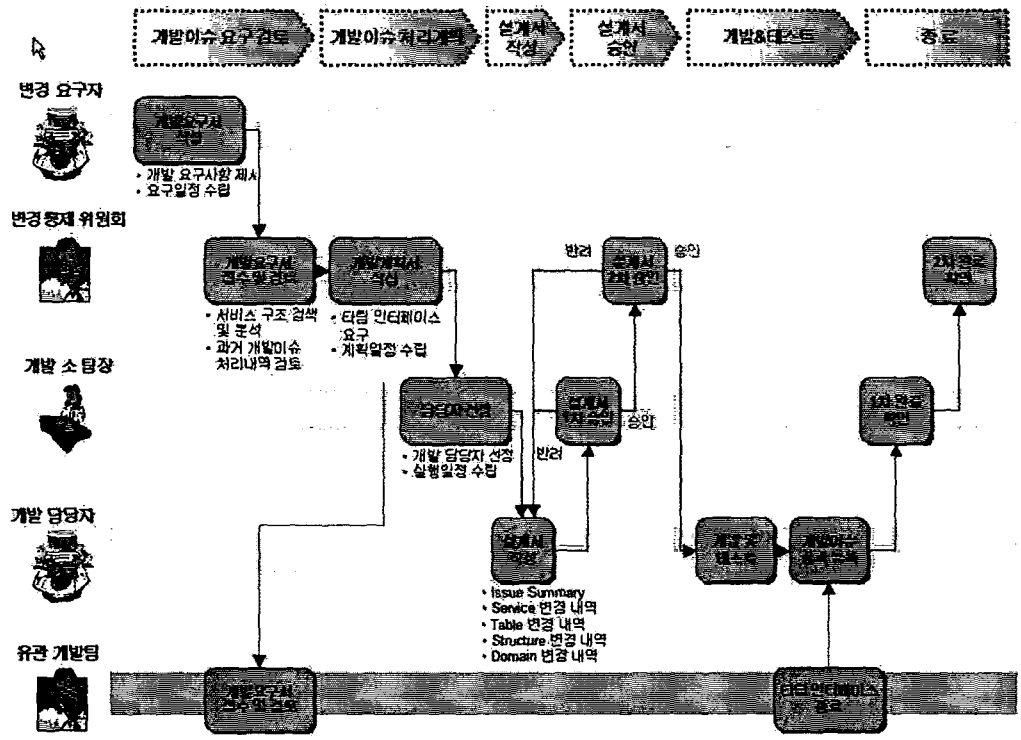
인공지능 분야에서는 기계학습(Machine Learning)을 이용한 변경영향분석 연구와 지식기반의 변경영향분석 연구가 있다. Shirabad et al. (2001)은 과거의 유지보수 작업 결과로부터 시스템 구성요소간의 관련성을 기계학습 기법을 이용하여 추출함으로써 변경에 대한 영향분석 방법론을 제안하였다. 이 방법은 호출그래프 및 프로그램 분할 기반의 영향분석 기법과는 달리 소스 코드의 정보만으로는 발견할 수 없는 의미적인 연관관계나 숨겨진 연관관계를 발견할 수 있다는 점에서 의의를 가지지만 영향분석의 정확성에 있어서 그 수준을 보장 할 수 없는 단점을 지닌다. 한편, 지식기반의 변경영향분석 연구의 대표적인 예로는 LaSSIE(Large Software System Information Environment)시스템을 들 수 있다[Devanbu et al., 1991; Devanbu and Jones, 1994]. LaSSIE 는 교환기 시스템 개발 시 개발자가 소프트웨어에 대한 정보를 쉽게 발견할 수 있도록 도메인 지식과 소스 코드 구조에 관한 정보를 결합하여 소프트웨어에 대한 정보를 의미적으로 검색할 수 있도록 지원한다. 그러나 복잡하고 잦은 변경이 요구되는 프로그램 호출 정보를 도메인 지식과 함께 지식으로 표현하여 관리함으로써 지식베이스에 대한 유지보수의 어려움이 논란의 대상이 되었다[Nardi and Brachman, 2003]. 또한

LaSSIE는 지식표현에 있어서 프레임 기반 모델을 사용함으로써 복잡한 계층구조의 지식표현에 적합하도록 설계하였으나, 추론 능력 측면에서 개념간 포함관계(Subsumption)에 대한 추론의 범주를 벗어나지 못 하고 있다[Devanbu, 1991; Devanbu and Jones, 1997; Nardi and Brachman, 2003]. 그러나, 영향분석 분야에서는 도메인 개념간의 복잡한 관계의 정의가 개념 계층분류 이상으로 중요하므로 개념간 포함관계 추론과 더불어 관계의 계층화(Super/Sub Property) 표현과 관계의 특성 및 제약을 고려한 추론(전이성에 대한 추론 등)이 동시에 중요시 된다. 이에 본 연구에서는 다양한 공리(Axiom)에 관한 표현력을 제공하는 OWL을 사용하여 개념간 포함관계에 대한 추론 외에 개념간 관계의 특성 및 제약에 기반한 추론을 통해 변경영향분석 영역에서의 한 차원 높은 의미적 검색 기능을 제시하고자 한다. 또한 잦은 변경이 필요한 프로그램의 호출 및 조립 구조에 대한 정보는 별도의 데이터 베이스에서 관리하도록 하고 도메인 지식에 대한 검색결과를 통해서 데이터베이스에 접근하는 2계층 구조로 시스템을 설계함으로써 LaSSIE의 단점으로 지적되는 지식베이스의 유지보수 문제를 최소화하고자 한다.

3. 소프트웨어 변경 영향분석 시스템의 요구사항

본 연구에서는 소프트웨어 변경 영향분석시스템에 대한 요구사항을 도출하기 위해 모 이동통신사의 과금/청구 시스템의 변경관리 업무를 조사하였다. 변경관리의 대상 시스템인 과금/청구 시스템은 3만 6천 여 개의 오브젝트로 구성되어 있고 한 개의 서비스에서 평균 7~10단계에 걸친 구성요소의 호출이 이루어지고 있는 기능간 상호의존도가 매우 높은 복잡한 시스템이다. 본 시스템은 크게 6개 기능으로 나누어져서 10 여 개의 협력사에 의해 유지보수 되고 있다. 이처럼 기능간 상호의존도와 응집도가 높은 소프트웨어를 여러 협력사에서 운영함으로써 변경 요청에 대한 시스템 영향 평가가 매우 중요시 된다. 또한 정보통신 관련 기술의 급속한 발전과, 법/제도의 변화, 차별화된 요금정책의 경쟁적인 도입 등으로 시스템 변경 요청 건수가 월 평균 100여 건에 달한다. 변경 작업의 실수가 장애로 연계되어 금전적 손실 및 대 고객 신뢰도 저하로 이어지기 때문에 엄격한 변경의 통제와 영향 평가 및 신속한 장애 원인 판단이 필수적이라 할 수 있다. [그림 1]은 과금/청구시스템의 유지보수 단계에서 변경통제 및 관리를 위한 절차를 제시하고 있다. [그림 1]에서와 같이 대규모 소프트웨어의 유지보수는 특성상 변경 작업이 장애로 이어지기 쉽기 때문에 모든 주요 활동에 대해서 2단계의 엄격한 승인 절차를 두고 있다. 또한 변경 작업을 통제하기 위해 변경통제 위원회를 구성하여 운영하고 있는데, 이는 각 업무분야별 풍부한 시스템 운영 경험을 보유한 책임자급들로 구성된다.

한편, 대상시스템의 구조측면에서 과금/청구 시스템은 특정 태스크처리를 위한 기본 단위인 서비스들로 구성된다. 각 서비스는 비즈니스 로직 계층의 각종 모듈(이하 PL이라 칭함)들과 데이터 계층의 각종 모듈(이하 DL 이라 칭함)들을 호출하여 실행 되며, PL 모듈은 또 다른 PL이나 DL모듈을 호출함으로써 평균 7~10 단계의 호출이 이루어진다. 데이터베이스의 테이블은 DL 모듈을 통해서 만 접근되며 DL 모듈은 데이터베이스 테



[그림 1] 소프트웨어 변경관리 프로세스

이블에 접근하기 위해서 필요한 논리적인 뷰의 변수 정의 파일을 불러서 사용하는데 이를 매크로라고 부른다. 상기의 변경관리 프로세스와 시스템 구조를 토대로 영향분석 시스템의 요구사항을 정리하면 다음과 같다.

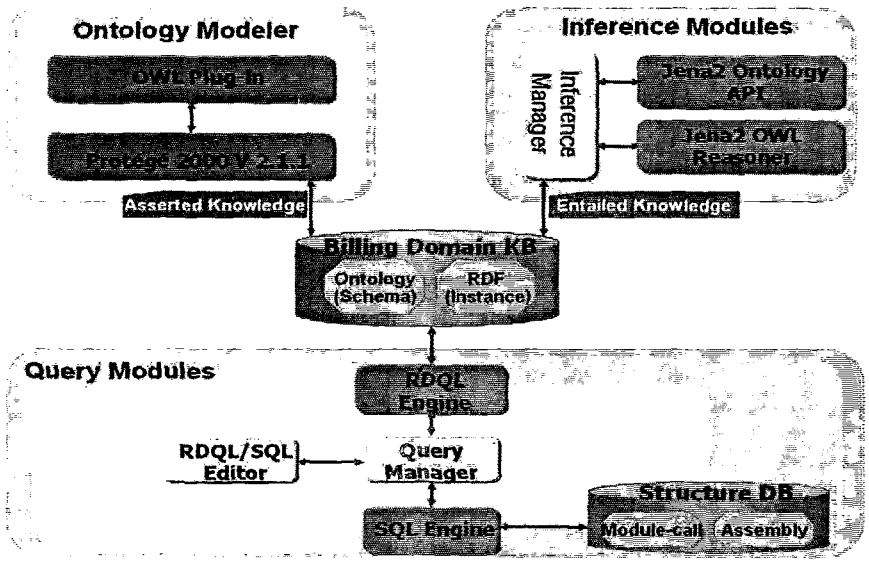
첫째, 변경 통제 위원회는 변경 요구사항에 대해 변경 대상 기능과 범위를 파악할 수 있어야 투입공수를 예측할 수 있으며 이를 토대로 변경 요구 부서에 조치기한을 피드백 해 줄 수 있다. 또한 어느 조직(서브팀)이 변경과 연관되어 있는지를 파악할 수 있어야 작업지시가 가능하다. 따라서 변경통제 위원회에서 영향분석 시스템에 요구하는 질의 내용은 업무적인 용어를 통해 해당되는 모듈을 찾는 질의와 담당 조직별 부하파악을 위한 질의가 중심이 된다. 둘째, 개발 소 팀장과 작업자는 변경작업을 효과적으로 수행하기 위해 모듈호출관계와 프로그램 조립구조에 대한 세부적인 영향 분석을 실시한다. 비록 개발자이더라도 다른 팀의 관할 모듈을 찾기 위해서는 비즈니스 업무용어를 통해 질의 하게 된다. 그러므로 변경 영향분석 시스템을 구현하기 위해서는 도메인 용어를 이용하여 모듈에 대한 의미적 검색이 가능하도록 모듈별 역할과 기능이 명시적인 온톨로지로 기술되어야 하고, 모듈간 호출 및 조립구조 관계에 대한 정의가 필요하며, 모듈별 담당조직과 담당자에 대한 정보가 기술되어야 함을 알 수 있다.

4. 온톨로지 기반 변경 영향분석 시스템(OntCIA)

4.1 전체 시스템 구조

OntCIA(Ontology based Change Impact Analysis System)는 [그림 2]와 같이 온톨로지 모델링 기능, 추론 기능, 질의 기능으로 구성된다. 온톨로지 모델링 기능(Ontology Modeler)은 과금/청구 도메인에서 사용되는 공유된 개념을 OWL 온톨로지로 표현하고 각 모듈의 기능을 RDF로 기술하도록 지원하며 이를 지식베이스에 저장 및 유지하는 기능을 수행한다. 이 때 온톨로지의 개발을 지원하는 도구로 Protégé 2000[3]을 활용한다. Protégé 2000은 온톨로지와 인스턴스를 개발하기 위한 그래픽 기반의 에디터를 제공하며 자체의 지식베이스를 갖추고 있어서 온톨로지 개발 시간을 단축할 수 있다. 또한 소스가 공개되어 있고 플러그인 모듈을 통한 확장구조를 제공하며 지식베이스에 접근할 수 있도록 자바기반의 API를 제공하므로 다양한 응용 프로그램과의 연계가 용이하다.

추론 기능(Inference Modules)은 지식베이스에 저장된 클래스(개념), 프로퍼티(개념간 관계 또는 속성)와 인스턴스에 대한 포함관계(Subsumption) 및 전이성(Transitivity) 등의 공리(Axiom)을 이용하여 추가적인 지식을 창출하는 기능을 수행하며 Jena 2[4]를 기반으로 하여 개발한다. Jena 2는 시맨틱 웹 응용프로그램을 구축하도록 지원하는 자바 기반의 프레임워크로써 RDF, RDFS, OWL, 추론엔진을 위한 프로그래밍 환경을 제공한다. 질의 기능(Query Modules)은 온톨로지와 RDF 인스턴스, 추론된 정보를 토대로 질의에 적합한 답변을 제공한다. RDF/XML



[그림 2] OntCIA 시스템 구조도

3) http://protege.stanford.edu/

4) http://jena.sourceforge.net

언어에 대한 질의를 위한 명시적인 언어로 SquishQL, RDQL, RQL 등이 있으나[Powers, 2003] 본 연구에서는 Jena에서 채택하고 있는 질의 언어인 RDQL을 이용하여 질의기능을 구성한다. 이때, RDQL 질의어를 온톨로지 정의내용을 모르는 사용자가 사용할 수 있도록 지원하기 위해서 RDQL 작성기(RDQL Editor)를 별도로 제공할 필요가 있다. 모듈간 호출 및 조립구조 정보는 유지보수 용이성을 위해 별도의 데이터베이스에서 관리하게 되며 RDQL 기반으로 비즈니스 도메인 용어를 통한 모듈검색이 이루어지면 탐색된 모듈명을 이용하여 호출 및 조립구조를 미리 정의된 SQL 템플릿을 사용하여 검색하게 된다. 다음 절에서는 각 기능별로 주요 설계 이슈와 해결책을 설명한다.

4.2 온톨로지 모델러

본 절에서는 온톨로지 모델링에 필요한 제반 이슈와 해결책을 중심으로 설명한다. 비즈니스 도메인에 관련된 온톨로지는 컴퓨터가 과금/청구 도메인의 업무용어를 이해함으로써 개발자가 아닌 관리자나 타 팀의 개발자가 업무용어를 통해 변경요청사항이 얼마나 많은 모듈의 수정과 관련되어 있는지를 질의할 수 있도록 하는 것을 목표로 한다. 예를 들면 관리자가 "복합상품 가입처리 기능" 이라는 질의에 대해 복합상품이 결합상품과 같은 용어이고, 가입처리는 "재가입"과 "신규가입"로 나누어 질 수 있다는 것을 이해하여 "결합상품 재가입 처리 모듈", "결합상품 신규가입 모듈", "복합상품 재가입 처리 모듈", "복합상품 신규가입 모듈" 까지를 검색해 주는 것을 말한다. 이 과정에서 결합상품이 복합상품과 같다는 것은 OWL의 equivalentClass 또는 sameAs axiom으로, 가입처리가 신규와 재가입으로 나누어지는 것은 subPropertyOf axiom을 통해 구현할 수 있다. 과금/청구 도메인 온톨로지의 개발 방법은 Noy와 McGuinness(2001)에 의해 개발된 온톨로지 개발 방법론에 의거하여 개발한다. 온톨로지 개발을 위한 주요 단계별 개발내용은 다음과 같다.

1) 온톨로지의 범위 결정

도메인 온톨로지의 범위를 결정하기 위해서는 Gruninger와 Fox(1995)가 제시한 Competency Question 방법이 많이 사용된다. 이 방법은 온톨로지를 포함하는 지식베이스가 답변해야 할 질문의 리스트를 미리 만들어 봄으로써 개발 범위와 지식표현의 수준을 결정하는 것이다. 이렇게 만들어진 질문리스트는 개발된 온톨로지를 평가하기 위한 리트머스 시험지의 역할을 하게 된다. 과금/청구 도메인에서 변경 영향분석의 목적으로 개발될 온톨로지는 <표 1>과 같은 질의에 답변할 수 있어야 한다.

<표 1> 과금/청구 도메인 온톨로지 개발을 위한 Competency Question

① 단말기 분납요금 계산과 관련된 모듈은 무엇인가?
② 결합상품 신규 가입자 등록 모듈이 호출하는 PL과 DL은 무엇인가?
③ 가입자에 대한 일지정지 처리 모듈을 호출하는 서비스는 무엇인가?
④ EBPP청구용 파일 생성 모듈을 담당하는 조직(사람)은 누구인가?
⑤ 싸이클 추출의 모든 후행 작업은 무엇이며 관련된 서비스는 무엇인가?
⑥ Macro_A 를 포함하는 모든 DL은 무엇인가?

상기 질문들을 분류해 보면 비즈니스 도메인 용어를 이용하여 모듈(PL, DL)을 찾는 질문(①)과 특정 모듈의 호출, 조립관계(⑥) 또는 담당자를 찾는 질문, 그리고 이 두 가지가 결합된 질문(②③④⑤)으로 나누어 진다. 따라서, 과금/청구 도메인 온톨로지의 범위는 비즈니스 도메인 개념과 이들의 관계에 대한 내용, 모듈별 관리 책임 주체에 대한 내용, 그리고 모듈간 호출 및 조립구조에 대한 내용으로 정의할 수 있다. 그러나, 이 중 모듈간 호출 및 조립구조에 대한 내용은 그 구조가 명료하며 정보의 획득 또한 자동화된 소스 스캔 프로그램에 의해 생성가능하며, 별도의 의미부여가 요구되지 않는 영역이다. 또한 대규모 소프트웨어의 경우, 호출 및 조립구조의 정보는 그 자체가 대 용량의 정보이어서 온톨로지로 표현하게 되면 추론의 성능 저하 문제와 유지 보수의 어려움이 발생한다. 이에 본 연구에서는 특정 모듈과 담당자(혹은 담당조직)를 찾기 위해 과금/청구 도메인에서 사용되는 공유된 개념과 관계만을 온톨로지의 개발 범위로 하고, 모듈간 호출 및 조립구조는 데이터베이스화 함으로써 추론의 성능문제와 유지보수문제를 동시에 해결하고자 한다.

2) 클래스와 프로퍼티의 정의 및 계층화

Competency Question 방법은 온톨로지의 범위를 정하는 데는 유용하게 사용되나 모든 질문을 빠짐없이 나열할 수 없기 때문에 누락되는 개념이 존재할 수 있으므로 진화적이고 반복적인 개발과정을 반드시 거쳐야 한다. 본 연구에서는 클래스와 프로퍼티 발견과정에서 누락을 최소화할 수 있도록 역 지식 공학 (Reverse Knowledge Engineering) 기법을 적용하여 클래스와 프로퍼티의 정의 및 계층화를 각각 수행하도록 한다. 현재 과금/청구 시스템의 각 소프트웨어 모듈은 해당 모듈의 수행 기능을 정의하는 한 라인에서 두 라인 정도의 Description을 헤더부분에 포함하고 있으며, 소스코드의 수정 시 동시에 해당 개발자에 의해 Description도 수정 되도록 규정화 되어 있다. 이러한 Description 정보는 업무용어를 사용하여 기술되어 있으며 다른 설계서나 분석서에 비해 정확한 소스코드에 대한 기능을 기술하고 있다. 따라서 이 정보를 샘플링과정을 통해 추출하여 주어(Subject), 술어(Predicate), 목적어(Object)의 구조(N3 구조)로 가공하고 각 개념과 술어에 대한 일반화 과정을 거쳐서 온톨로지를 완성할 수 있다. N3 구조의 명시적인 형태로 프로그램의 Description을 가공하는 과정에서 효과적인 지식의 표현과 추론을 위해서는 각 개념과 술어가 최대한 공유될 수 있도록 하는 것이 중요하다. 개념의 공유를 위해서는 Description 상의 주어와 목적어를 대상으로 유사개념 과 동의이음 및 동음이의 관계를 판단하는 용어의 분류 및 표준화가 동시에 진행될 경우 더욱 효과적이다. 술어의 공유를 위해서 Description의 패턴을 분석해 보면 "Role"과 "Target"으로 구분이 됨을 알 수 있다. "Role"은 "~을 ~한다" 형태의 기능을 기술하는 부분이며, "Target"은 "~을 대상으로" 형태의 기술을 말한다. 따라서, "hasRole"과 "target"을 기본 술어로 정하여 N3 구조를 추출할 수 있다. <표 2>는 Description을 통해 N3 구조를 추출한 예이다.

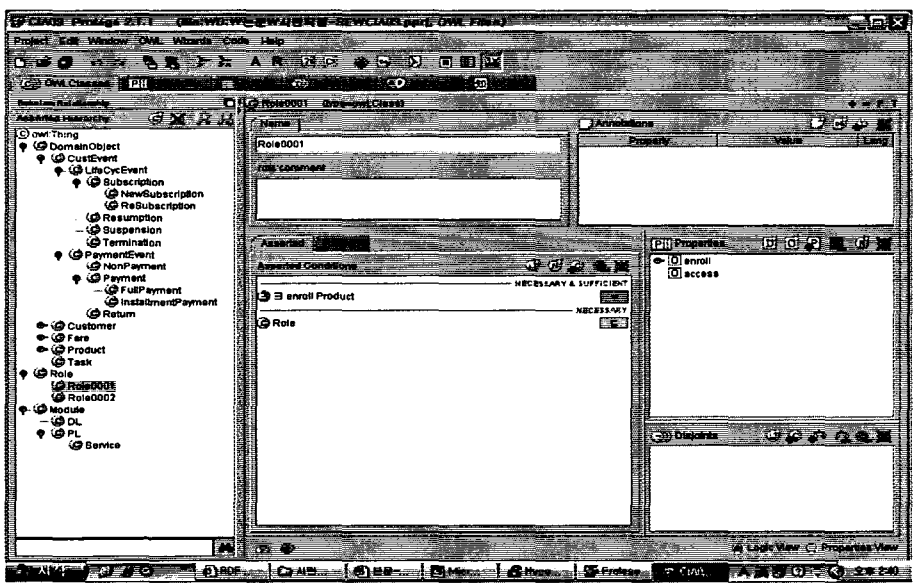
<표 2>에서 "cia"는 과금/청구 도메인 온톨로지의 네임스페이스(Namespace)에 대한 접두어로써 다른 도메인 온톨로지의 용어와 구별하기 위한 것이고, "???"는 익명의 자원(Anonymous resource)에 대한 표시로써 "Role" 클래스의 하위 클래스이다. 상기 예에서 "hasRole" 속성을

<표 2> Description으로부터 N3구조의 추출 예

Description	usr/pp_bill/pp0101.c 는 결합상품 (iComboProduct)에 신규고객 (iNewCust)을 등록한다.		
N3 표현	cia:usr/pp_bill/pp0101.c	cia:target	cia:iComboProduct .
	cia:usr/pp_bill/pp0101.c	cia:hasRole	??? .
	???	cia:enroll	cia:iNewCust .

사용하지 않고 N3 구조를 만들면 더 단순한 구조를 만들 수 있으나, "Role" 클래스에 대한 Redundancy가 예상 되므로 온톨로지 정규화 차원에서 이를 별도로 분리 하기로 한다. 온톨로지 정규화는 온톨로지의 재사용과 유지보수 용이성을 달성하기 위한 모듈화의 과정으로써 데이터베이스의 정규화 과정과 비교된다[Rector, 2003; Gu et al., 1999; Rector et al., 1999].

이렇게 N3 구조로 도메인 용어가 추출되면 클래스에 대해 각각 is-A 관계의 계층화를 하게 된다. 즉, 클래스 X의 상위 클래스가 클래스 Y라면 클래스 X는 클래스 Y의 서브 클래스가 되어 클래스 Y의 모든 속성을 상속 받게 된다. Is_A 관계의 계층화는 전이적(Transitive) 특성을 가지므로 계층화를 통해 다양한 형태의 추론이 가능해 지는 장점을 가진다. Uschold와 Gruninger (1996)에 의하면 Class 계층화를 위한 방법으로 세가지 접근법을 제시하고 있다. 첫째, 하향식 접근법은 일반적인 개념에서 출발하여 특수화(Specialization) 과정을 통해 계층화를 하고 둘째, 상향식 접근법은 구체적인 개념에서 출발하여 일반화(Generalization) 과정을 거쳐서 계층화하며 셋째, 혼합식 접근법은 중요한 개념을 중심



[그림 3] 과금/청구 도메인의 클래스 계층화

으로 일반화와 특수화를 병행함으로써 계층화를 한다. 본 연구에서는 역 지식공학 기법의 적용 특성 상 구체적인 개념에서 출발하여 일반화의 과정을 거침으로써 계층화를 하는 것이 바람직하다. 일반화 과정을 통해서 과금/청구 도메인에서 추출된 상위 클래스는 [그림 3]에서와 같이 고객(Customer), 상품(Product), 요금(Fare), 고객이벤트(CustEvent), 작업(Task), 모듈(Module), 역할(Role)로 정의 되었다.

프로퍼티(Property)도 동일한 방식으로 계층화하여 상위 프로퍼티(Super Property)와 하위 프로퍼티(Sub Property)를 정의할 수 있다. 과금/청구 도메인의 경우, 다양한 개념간 관계에 대한 용어를 기술할 필요가 있기 때문에 상하위 프로퍼티의 정의가 매우 중요시 된다. 예로써, "접근한다" 라는 프로퍼티를 "등록한다" 와 "갱신한다"의 상위 프로퍼티로 정의하면 사용자가 "등록한다" 라는 용어를 사용하여 질의 하더라도 추론과정을 통해 "접근한다"의 역할을 갖는 모듈도 검색에 포함하게 된다. 프로퍼티의 정의와 계층화가 이루어지면 각 프로퍼티에 대한 특성과 제약을 정의하게 되는데 이는 추론 기능과 밀접한 관련이 있으므로 추론부문에서 설명한다.

3) 인스턴스의 기술

개발된 OWL 온톨로지를 준수하여 모든 모듈의 헤더 부분에 존재하는 Description을 RDF형식의 인스턴스로 개발한다. 이 과정은 Protégé 2000의 Individual 탭을 이용하여 직접 등록 할 수도 있고 N3 표현으로 작성하여 RDF 포맷으로

<표 3> 과금/청구 도메인의 OWL온톨로지와 RDF 인스턴스의 일부

OWL 온톨로지

```
<?xml version="1.0"?>
<rdf:RDF xmlns=http://www.ariosoft.com/hssong/CIAont#
  xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
  xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.ariosoft.com/hssong/CIAont">
<owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="DL">
      <owl:disjointWith>
          <owl:Class rdf:about="#PL"/>
      </owl:disjointWith>
      <rdfs:subClassOf>
          <owl:Class rdf:ID="Module"/>
      </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="update">
      <rdfs:range rdf:resource="#DomainObject"/>
      <rdfs:subPropertyOf rdf:resource="#access"/>
</owl:ObjectProperty>
</rdf:RDF>
```

RDF 인스턴스

```
<rdf:RDF>
  <NewCustomer rdf:ID="iNewCustomer"/>
  <Role0001 rdf:ID="iRole0001">
    <enroll rdf:resource="#iNewCustomer"/>
  </Role0001>
  <PL rdf:about="#usr/pp_bill/pp0101.c">
    <hasRole rdf:resource="#iRole0001"/>
  </PL>
</rdf:RDF>
```

변환할 수도 있다. <표 3>은 과금/청구 도메인의 OWL 온톨로지와 RDF 인스턴스의 일부이다. <표 3>에서는 "DL" 클래스가 "PL" 클래스와 "disjointWith" (DL클래스의 맴버는 PL클래스의 Member가 될 수 없음) 관계에 있고, "Module" 클래스의 하위 클래스 임을 보여 주고 있다. 또한 "update" 프로퍼티는 "DomainObject" 클래스의 멤버를 속성 값으로 가지며, "access" 프로퍼티의 하위 프로퍼티 임을 보여 주고 있다. RDF 인스턴스에 기술된 내용은 "usr/pp_bill/pp0101.c"가 "PL" 클래스의 모듈이며 신규고객("iNewCustomer")을 등록("enroll")하는 역할을 수행함을 기술하고 있다.

4.3 추론 모듈

본 절에서는 OWL의 다양한 공리(Axiom) 정의를 토대로 과금/청구 도메인에서 어떠한 추론이 가능한지를 소개한다.

1) subClassOf Axiom을 이용한 추론 예

OWL의 subClassOf Axiom 정의를 이용하면 개념과 인스턴스에 대한 포함관계와 분류에 대한 추론이 가능하다. 표 4에서는 고객가입을 승인하는 모듈을 검색하는 질의에 대해 고객가입만이 아닌 신규가입과 재가입을 승인하는 모듈도 검색하는 추론을 예로 보이고 있다. OWL 온톨로지는 "Subscription"의 서브클래스로 "ReSubscription"과 "NewSubscription"이 있음을 정의하고 있다. 이 정의에 기반한 추론을 통해 모듈 "#usr/pp_bill/pp0101.c"이 비록 RDF 인스턴스 부문에서는 "ReSubscprition"을 승인하는 역할로 정의되어 있지만 "Subscription"을 승인하기도 한다는 사실을 추론을 통해 보여 주고 있다.

<표 4> subClassOf Axiom을 이용한 추론 예

질의	고객가입을 승인하는 모듈은 무엇인가?
OWL 온톨로지	<owl:Class rdf:ID="ReSubscription"> <rdfs:subClassOf rdf:resource="#Subscription"/> </owl:Class> <owl:Class rdf:ID="NewSubscription"> <rdfs:subClassOf rdf:resource="#Subscription"/> </owl:Class>
RDF 인스턴스	<cia:PL rdf:about="#usr/pp_bill/pp0101.c"> <cia:hasRole> <cia:Role rdf:ID="role0011"> <cia:confirm> <cia:Subscription rdf:ID="iReSubscription"/> </cia:confirm> </cia:Role> </cia:hasRole> </cia:PL>
추론된 사실	<cia:PL rdf:about="#usr/pp_bill/pp0101.c"> <cia:hasRole> <cia:Role rdf:ID="role0011"> <cia:confirm> <cia:ReSubscription rdf:ID="iSubscription"/> </cia:confirm> </cia:Role> </cia:hasRole> </cia:PL>

2) subPropertyOf Axiom을 이용한 추론 예

OWL의 표현력을 높이는 요소 중 하나가 프로퍼티에 관한 계층구조를 정의할 수 있다는 것이다. 그러나 클래스와는 달리 하위 프로퍼티가 모든 부모 프로퍼티의 특성을 상속받는 것은 아니다. 즉, 상위 프로퍼티가 전이성을 가진다고 하여 하위 프로퍼티도 전이성을 갖는 것은 아니다. 표 5는 단말기 요금정보에 접근하는 DL을 검색하는 질의에 대한 추론 결과를 보여주고 있다. 온톨로지에서 정의부문에서 "access"의 하위 프로퍼티로 "inquire"와 "update"가 있음을 기술하고 있기 때문에 비록 DL 모듈 "#usr/pp_bill/pd0110.c"가 RDF 인스턴스 영역에서 단말기 요금정보를 "inquire" 하는 것으로 정의되어 있어도 이는 "access"의 관계도 있음을 추론을 통해 보여주고 있다.

3) TransitiveProperty Axiom을 이용한 추론 예

특정 프로퍼티가 전이성(Transitivity)을 가지도록 정의되면 클래스와 인스턴스들 간의 다양한 전이적 추론이 가능해 진다. <표 6>은 Bill Production의 선행 작업을 모두 검색하는 질의에 대한 추론 결과를 보여 주고 있다. 온톨로지 부분에서는 "followedBy" 속성을 TransitiveProperty로 선언하였다. 인스턴스 부분에서는 "iBillProduction"의 선행작업이 "iRerating", "iRerating"의 선행작업이 "iReratingCheck", "iReratingCheck"의 선행작업이 "iCycleExtraction" 작업임을 기술하고 있다. 따라서 추론된 사실에서는 "iCycleExtraction"도 "iBillProduction"의 선행작업임을 보여주고 있다.

<표 5> subPropertyOf Axiom을 이용한 추론 예

질의	단말기 요금정보를 접근 하는 DL을 검색 하라.
OWL 온톨로지	<owl:ObjectProperty rdf:ID="update"> <rdfs:subPropertyOf rdf:resource="#access"/> </owl:ObjectProperty> <owl:ObjectProperty rdf:ID="inquire"> <rdfs:subPropertyOf rdf:resource="#access"/> </owl:ObjectProperty>
RDF 인스턴스	<cia:DL rdf:about="#usr/pp_bill/pd0110.c"> <cia:hasRole> <cia:Role rdf:ID="role0012"> <cia:inquire> <cia:HanSetFare rdf:ID="iHandSetFare"/> </cia:inquire> </cia:Role> </cia:hasRole> </cia:DL>
추론된 사실	<cia:DL rdf:about="#usr/pp_bill/pd0110.c"> <cia:hasRole> <cia:Role rdf:ID="role0012"> <cia:access> <cia:HanSetFare rdf:ID="iHandSetFare"/> </cia:access> </cia:Role> </cia:hasRole> </cia:DL>

<표 6> TransitiveProperty Axiom을 이용한 추론 예

질의	Bill Production의 선행작업을 검색 하라.
OWL 온톨로지	<owl:TransitiveProperty rdf:ID="followedBy"> <rdfs:range rdf:resource="#Task"/> <rdfs:domain rdf:resource="#Task"/> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/> </owl:TransitiveProperty>
RDF 인스턴스	<cia:Task rdf:ID="iBillProduction"> <cia:followedBy> <cia:Task rdf:ID="iRerating"> <cia:followedBy> <cia:Task rdf:ID="iReratingCheck"> <cia:followedBy> <cia:Task rdf:ID="iCycleExtraction"/> </cia:followedBy> </cia:Task> </cia:followedBy> </cia:Task> </cia:followedBy> </cia:Task>
추론된 사실	<cia:Task rdf:ID="iBillProduction"> <cia:followedBy> <cia:Task rdf:ID="iCycleExtraction"/> </cia:followedBy> </cia:Task>

4) 프로퍼티 제약식을 이용한 클래스간 포함관계(SubSumption) 추론 예

프로퍼티에는 전이성 같은 특성(Characteristic)부여 외에도 다양한 제약(Constraint)을 부여하여 추론에 활용할 수 있다. <표 7>은 클래스 "Role0002"가 "Role0001"의 서브클래스임을 추론하는 예이다. 온톨로지 정의부문을 보면 "Combo"가 "Product" 클래스의 서브클래스임이 정의되어 있다. 또한 "Role0001" 클래스의 모든 멤버는 최소한 하나 이상의 "Product" 클래스에 enroll 되어야 함을 정의하고 있다. 또한 "owl:equivalentClass" 정의가 되어 있으므로 역으로 모든 "Product" 클래스에 하나이상 "enroll" 되는 멤버이면, 이는 "Role0001" 클래스의 멤버임을 나타내기도 한다. 그리고 "Role0002" 클래스의 모든 멤버는 최소한 하나 이상의 "Combo" 클래스에 "enroll" 되어야 함을 정의하고 있다. 또한 "owl:equivalentClass" 정의가 되어 있으므로 그 역도 성립한다. 그런데, "Combo" 클래스는 "Product" 클래스의 서브클래스이므로 결국 "Role0002"는 "Role0001"의 서브클래스임을 추론할 수 있다. 이를 술어 논리를 이용하여 표현하면 다음과 같다.

$Role0001 \equiv \exists\, enroll.(Product)$
$Role0002 \equiv \exists\, enroll.(Combo)$
$Combo \subseteq Product$ 로 부터

추론된 사실 $Role0001 \supseteq Role0002$

<표 7> 프로퍼티 제약식을 이용한 클래스간 포함관계(SubSumption) 추론 예

질의	Role 간의 SubSumption 관계를 분석하라
OWL 온톨로지	<owl:Class rdf:ID="Product"/> <owl:Class rdf:ID="Combo"> <rdfs:subClassOf rdf:resource="#Product"/> </owl:Class> <owl:ObjectProperty rdf:ID="enroll"> <rdfs:range rdf:resource="#DomainObject"/> <rdfs:domain rdf:resource="#Role"/> </owl:ObjectProperty> <owl:Class rdf:ID="Role0001"> <rdfs:subClassOf rdf:resource="#Role"/> <owl:equivalentClass> <owl:Restriction> <owl:onProperty> <owl:ObjectProperty rdf:about="#enroll"/> </owl:onProperty> <owl:someValuesFrom rdf:resource="#Product"/> </owl:Restriction> </owl:equivalentClass> </owl:Class> <owl:Class rdf:ID="Role0002"> <rdfs:subClassOf rdf:resource="#Role"/> <owl:equivalentClass> <owl:Restriction> <owl:someValuesFrom rdf:resource="#Combo"/> <owl:onProperty> <owl:ObjectProperty rdf:about="#enroll"/> </owl:onProperty> </owl:Restriction> </owl:equivalentClass> </owl:Class>
추론된 사실	<owl:Class rdf:ID="Role0002"> <rdfs:subClassOf rdf:resource="#Role0001"/> </owl:Class>

프로퍼티 제약표현에 의한 Role 간 포함관계 추론은 과금/청구 도메인의 지식베이스 유지 보수 작업에 매우 유용하게 사용될 수 있다. 개발자가 특정 모듈을 변경함에 따라 Description을 수정하게 되면 지식베이스에 기술된 Role 관련 RDF 인스턴스를 수정해야 한다. 그러나 본 도메인의 모듈 수가 많기 때문에 수 많은 Role 클래스 중에서 적절한 Role을 찾아서 수정하기는 쉽지 않다. 추론을 통해서 Role 트리가 만들어지면 특정 모듈의 Role에 대한 수정작업이 한층 용이해진다.

Jena 2는 도메인 지식의 표현형식에 따라 RDFS 추론기(RDFS rule reasoner), OWL 추론기(OWL FB reasoner), DAML 추론기(DAML miro reasoner), 규칙 추론기(Generic rule reasoner) 등의 다양한 추론기를 지원한다. 본 연구에서는 OWL을 이용하여 지식을 표현하고 있

기 때문에 OWL 추론기를 이용하여 추론을 수행한다. [그림 4]는 Jena의 OWL 추론기를 호출하여 <표 7>에 나타나있는 추론을 수행하는 자바코드와 실행결과를 보여주고 있다. [그림 4]의 라인 7에서 라인 11까지는 과금/청구 OWL 온톨로지에 특화된 OWL 추론기(OWL reasoner)의 인스턴스를 생성하여 RDF 인스턴스에 적용함으로써 추론모델을 생성하는 부분이다. 라인 12에서 라인 14는 생성된 추론모델에서 "Role0002"를 주어로 가지는 모든 RDF 인스턴스를 출력하는 부분이다. 실행결과에서는 "Role0002"가 "Role0001"의 서브클래스라는 결과도 포함되어 있음을 알 수 있다.

OWL 추론기를 이용한 추론 프로그램 예

```
01  import com.hp.hpl.jena.rdf.model.*;
02  import com.hp.hpl.jena.util.*;
03  import com.hp.hpl.jena.reasoner.*;
04
05  public class owltest02 extends Object {
06      public static void main (String args[]) {
07              Model schema = ModelLoader.loadModel("file:CIA03.owl");
08              Model data = ModelLoader.loadModel("file:CIA03.rdf");
09              Reasoner reasoner = ReasonerRegistry.getOWLReasoner();
10              reasoner = reasoner.bindSchema(schema);
11              InfModel infmodel = ModelFactory.createInfModel(reasoner, data);
12              Resource Role0002 = infmodel.getResource("http://a.com/ontology#Role0002");
13              System.out.println("Role0002 *:");
14              printStatements(infmodel, Role0002, null, null); }
15      public static void printStatements(Model m, Resource s, Property p, Resource o) {
16              for (StmtIterator i = m.listStatements(s,p,o); i.hasNext(); ) {
17                  Statement stmt = i.nextStatement();
18                  System.out.println(" - " + PrintUtil.print(stmt)); } }
19  }
```

추론 결과

```
---------- Java Execution.. ----------
Role0002 *:
 - (http://a.com/ontology#Role0002 rdfs:subClassOf b8bef7:fe04630329:-8000)
 - (http://a.com/ontology#Role0002 rb:prototype b8bef7:fe04630329:-7fea)
 - (http://a.com/ontology#Role0002 rdfs:subClassOf http://a.com/ontology#Role0002)
 - (http://a.com/ontology#Role0002 rdfs:subClassOf owl:Thing)
 - (http://a.com/ontology#Role0002 rdf:type rdfs:Class)
 - (http://a.com/ontology#Role0002 rdfs:subClassOf rdfs:Resource)
 - (http://a.com/ontology#Role0002 rdf:type owl:Class)
 - (http://a.com/ontology#Role0002 rdfs:subClassOf http://a.com/ontology#Role)
 - (http://a.com/ontology#Role0002 owl:equivalentClass b8bef7:fe04630329:-8000)
 - (http://a.com/ontology#Role0002 rdf:type rdfs:Resource)
 - (http://a.com/ontology#Role0002 rdfs:subClassOf b8bef7:fe04630329:-7fff)
 - (http://a.com/ontology#Role0002 rdfs:subClassOf http://a.com/ontology#Role0001)
 - (http://a.com/ontology#Role0002 owl:equivalentClass http://a.com/ontology#Role0002)
Normal Termination
출력 완료 (3초 경과).
```

[그림 4] 추론 프로그램 예와 추론 결과

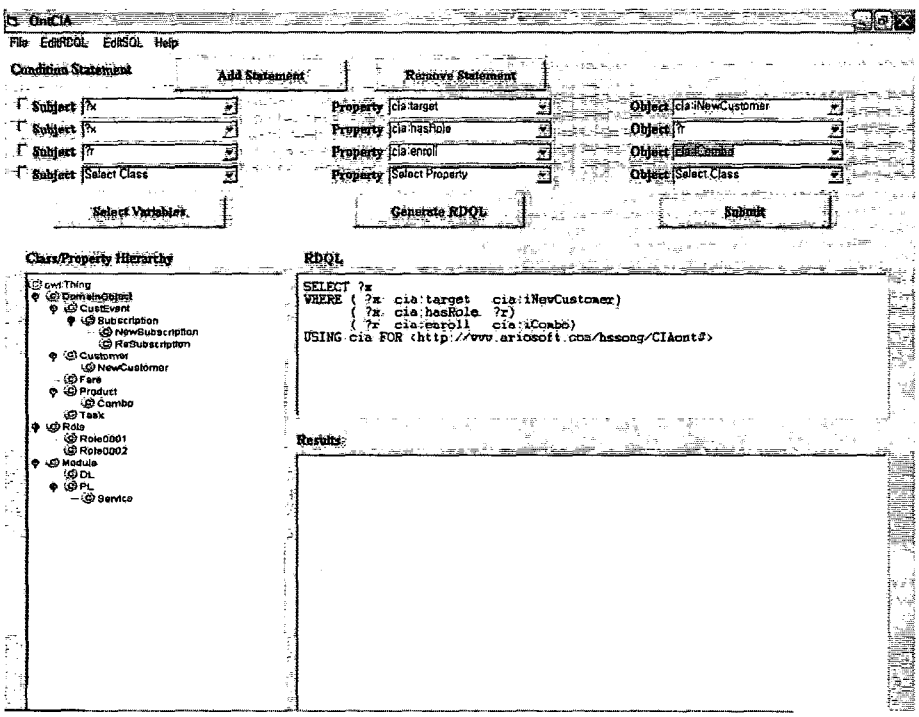
4.4 질의 모듈

질의모듈은 비즈니스 도메인 용어를 이용하여 RDQL을 기반으로 해당 모듈을 찾는 부분과 발견된 모듈을 입력받아 SQL을 이용하여 호출 및 조립구조를 탐색하는 부분으로 구성된다. <표 8>은 RDQL을 사용하여 "신규고객을 결합상품에 등록하는 모듈은 무엇인가?"를 표현하고 있다.

그러나 과금/청구 도메인의 클래스와 프로퍼티를 모르는 사용자는 위와 같은 질의어를 직접 작성한다는 것이 어렵다. 따라서 RDQL 작성을 위한 별도의 작성기 화면이 제공되어야 한다. [그림 5]는 RDQL 작성기(RDQL Editor) 화면을 예로 보이고 있다. 이 화면에서는 질의의 대상이 되는 주어와 목적어는 온톨로지의 클래스 계층도를 통해 사용자가 선택할 수 있도록 지원하며 선택된

<표 8> RDQL 검색 예

```
SELECT    ?x
WHERE  ( ?x  cia:target      cia:iNewCustomer)
       ( ?x cia:hasRole      ?r )
       ( ?r cia:enroll       cia:iCombo)
USING cia FOR <http://www.ariosoft.com/hssong/CIAont#>
```



[그림 5] RDQL 작성기 화면

클래스에 적용되는 프로퍼티만을 브라우저해 줌으로써 술어의 선택도 가이드 하게 된다.

한편 모듈간 호출구조와 조립구조의 경우, 추론의 성능문제와 유지보수 용이성을 고려하여 별도의 데이터 베이스에서 관리하게 된다. 따라서 "신규고객을 결합상품에 등록하는 모듈이 호출하는 모든 모듈은 무엇인가?" 와 같이 결합된 질의의 경우, 먼저 <표 8>과 같이 "신규고객을 결합상품에 등록하는 모듈은 무엇인가?"에 해당하는 모듈을 검색한 후 SQL문을 사용하여 호출하는 모듈들에 대한 정보를 데이터 베이스에서 직접 검색할 필요가 있다. 이때, SQL문을 자동 생성하기 위해서 특정 모듈을 기준으로 호출관계에 대한 순전개(Explosion) 및 역전개(Implosion), 조립관계에 대한 순전개 및 역전개 등 네 가지의 SQL 템플릿을 미리 만들어 두어 RDQL의 실행과 동시에 수행 되도록 할 수 있다. 호출 및 조립구조에 대한 정보는 초기에 소스코드 스캐닝 과정을 거쳐서 만들어져서 데이터 베이스에 적재(Load)되며, 이후 형상관리 시스템에 의해 프로그램의 형상 변경이 이루어질 때마다 실시간으로 데이터 베이스를 갱신하는 구조를 취한다. 따라서 호출 및 조립 구조 변경에 대한 별도의 유지 보수 노력은 필요가 없게 된다.

한편, 프로그램 수정에 따라 개발자에 의해 헤더 부분에 작성되는 Description이나 담당자 정보가 수정되면 개발자는 이를 온톨로지 관리시스템에 반영하여야 한다. Description의 변경은 온톨로지 스키마의 변경을 요구하기 보다는 RDF 인스턴스 수준의 추가 또는 수정에 해당한다. 이는 특정 모듈에 대해 Role을 변경하는 수준에서 이루어지므로 Role 트리를 이용하여 개발자가 Role을 추가하거나 수정하는 간단한 작업을 통해 이루어 질 수 있다.

5. 결론

본 연구는 RDF와 온톨로지 등의 시맨틱 웹 기술을 이용하여 대규모 소프트웨어의 변경영향분석에 이슈가 되는 복잡성과 비가시성 문제를 해결하고자 하였으며 이를 위해 온톨로지 기반의 변경영향 분석 시스템(OntCIA)을 제안하였다. 본 연구는 다음과 같은 점에서 의의를 가진다. 첫째, 도메인 지식과 소스 코드의 구조적인 정보를 결합하여 의미적 검색이 가능한 변경 영향 분석 시스템을 제안하였다. 둘째, 최근 W3C에서 권고안으로 채택되고 풍부한 표현력을 지닌 OWL을 기반으로 하여 과금/청구 도메인의 온톨로지를 역지식공학 방법에 의해 설계하였고 변경 영향분석 분야에서 활용될 수 있는 다양한 추론 기능들을 제시하였다. 셋째, 잦은 변경이 요구되는 호출 및 조립구조 정보는 데이터 베이스에서 관리하고 도메인 지식은 온톨로지로 관리함으로써 유지 보수가 용이한 시스템 구조를 제안하였다.

본 연구에서는 소스 프로그램 내에 기술된 Description정보를 토대로 역 지식공학 방식에 의해 온톨로지를 개발하였다. 이를 위해 OWL의 제약식과 공리(Axiom) 표현능력을 주로 활용하였다. 그러나, 소프트웨어 변경 업무 수행과 관련된 개발자들의 경험과 지식이 추출되어 동시에 반영될 수 있다면 소프트웨어 변경영향분석 작업이 한층 용이해 질 것이다. 따라서, 과금/청구 도메인에서 소프트웨어 변경영향분석을 위한 규칙의 습득과 표현이 동시에 이루어지는 시스템으로의 발

전이 필요하다. 개발자의 경험 및 지식에서 추출된 규칙이 적용될 수 있는 예로는 변경 요구사항에 종속되는 의미적 영향관계 평가를 포함한다. 즉 신규 할인 상품 적용에 따른 소스코드 변경 요구의 경우, 요금 계산 기능도 변경 대상이 되겠지만, 이와는 전혀 구조적인 관계를 갖지 않는 고객 정보관리 기능에 대한 변경도 동시에 이루어 져야 한다. 본 연구에서 제시하고 있는 변경 영향평가 시스템은 이 부분은 사람이 하는 것으로 가정하고 있으나, 규칙의 적용을 통해 어느 정도 자동화가 가능할 것으로 판단한다. 시맨틱 웹 기술은 구성원들에게 공유된 개념을 명시적으로 표현하는 수단을 제공함으로써 새로운 참여자를 포함하는 구성원간 지식전이를 용이하게 한다 점에서 실무적으로도 그 중요성이 부각될 것으로 기대한다. 특히, 대규모 소프트웨어의 변경관리 분야처럼 업무 처리의 높은 신뢰성이 요구되며 구성원간 의사소통이 많이 요구되는 분야일수록 활용효과가 증대할 것이다.

참고문헌

전종홍, 이원석, 이강찬, "시맨틱 웹 기술을 적용한 지식관리시스템 아키텍처에 관한 연구", 한국전자거래학회지, 8권 4호(2003), 183-205.

조성정, 김진형, "시맨틱 웹의 응용사례연구", 정보과학회지, 21권 3호(2003), 11-17.

최중민, "시맨틱 웹의 개요와 연구동향", 정보과학회지, 21권 3호(2003), 4-10.

Arnold, R. S., and S. A. Bohner, "Impact Analysis-towards a framework for comparison", In Proceedings of the International Conference on Sofrware Maintenance, 1993, pp.292-301.

Berners-Lee, T., J. Hendler, and O. Lassila, "The Semantic Web", Scientific American, Vol.284, No.5(2001), 34-43.

Bohner, S. A., and R. S. Arnold, Sofeware Change Impact Analysis, IEEE Computer Society Press, 1996.

Brooks, F. P., "No Silver Bullet: Essence and Accidents of Software Engineering", IEEE Computer Magazine, 1987.

Cost, R. S., T. Finin, A. Joshi, Y. Peng, C. Nicholas, I. Soboroff, H. Chen, L. Kagal, F. Perich, Y. Zou, and S. Tolia, "ITtalks: A case study in the semantic web and DAML+OIL", IEEE Intelligent Systems, Vol.17, No.1(2002), 40-47.

Daconta, M. C., L. J. Obrst, and K. T. Smith, The Semantic Web, Wiley Publishing, 2003.

Davies, J., D. Fensel, and F. V. Harmelen, Towards the Semantic Web: Ontology-driven Knowledge Management, John Wiley & Sons, 2003.

Devanbu, P., R. J. Brachman, P. G. Selfridge, and B. W. Ballard, "LaSSIE: a Knowledge-Based Software Information System", International Conference on Software Engineering, 1991.

Devanbu, P. T., and M. A. Jones, "The Use of Description Logics in KBSE systems", ACM Transactions on Software Engineering and Methodology, Vol.6, No.2 (1997), 141-172.

Gruber, T., "A translation approach to portable ontologies", Knowledge Acquisition, Vol.5, No.2(1993), 199-220.

Gruninger, M., and M. S. Fox, "Methodology for the design and evaluation of ontologies", In Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI-95), 1995.

Gu, H. H., Y. Perl, J. Geller, M. Halper, and M.

Singh, "A methodology for partitioning a vocabulary hierarchy into trees", Artificial Intelligence in Medicine, Vol.15(1999), 77-98.

Kellens, A., Using Inductive Logic Programming to Derive Software Views, Ph. D. Dissertation, 2003.

M. Klein, and U. Visser, "Guest Editors' Introduction: Semantic Web Challenge 2003", IEEE Intelligent Systems, Vol.19, No.3(2004), 31-33.

Law, J., and G. Rothermel, "Whole Program Path-Based Dynamic Impact Analysis", In Proceeding ICSE , 2003, 308-318.

Michalowski, M., J. L. Ambite, C. A. Knoblock, S. Minton, S. Thakkar, and R. Tuchinda, "Retrieving and Semantically Integrating Heterogeneous Data from the Web", IEEE Intelligent Systems, Vol.19, No.3(2004), 72-79.

Nardi, D., and R. J. Brachman, An Introduction to Description Logics, Description Logic Handbook 2003, Cambridge University Press, 5-44.

Noy, N., and D. L. McGuinness, Ontology Development 101: A Guide to Creating Your First Ontology, http://www.co-ode.org/resources/.

Powers, S., Practical RDF, O'reilly, 2003.

Rector, A. L., "Modularisation of Domain Ontologies Implemented in Description Logics and related formalisms including OWL", In Proceeding KCap2003, 2003.

Rector, A., N. Drummond, M. Horridge, J. Togers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe, OWL Pizzas: Practical Experience of Teaching OWL-DL, http://www.co-ode.org/resources/.

Rector, A. L, P. E. Zanstra, W. D. Solomon, J. E. Rogers, R. Baud, W. Ceusters, W. Claassen, J. Kirby, J. M. Rodrigues, A. R. Mori, E. Haring, and J. Wagner, "Reconciling Users' Needs and Formal Requirement", IEEE Tran on Information Technology in BioMedicine, Vol.2(1999), 229-242.

Reynolds, D., S. Cayzer, and I. Dickinson, SWAD-Europe Deliverable 12.1.1: Semantic web applications-analysis and selection, http://www.w3.org/2001/sw/Europe/reports/open_demonstrators/hp-applications-selection.html.

Ryder, B. G., and F. Tip, "Change impact analysis for object-oriented programs", In Proceeding PASTE, 2001, 46-53.

Shirabad, J., S. Timothy, C. Lethbridge, and S. Matwin, "Supporting Software Maintenance by Mining Software Update Records", In Proceeding IEEE International Conference on Software Maintenance (ICSM'01), 2001

Uschold, M., and M. Gruninger, "Ontologies: Principles, Methods and Applications", Knowledge Engineering Review, Vol.11, NO.2(1996), 93-136.

Zhao, J., H. Yang, L. Xiang, and B. Xu, "Change impact analysis to support architectural evolution", Journal of Software Maintenance, Vol.14, No.5(2002), 317-333.

Abstract

OntCIA: Software Change Impact Analysis System Based on the Semantic Web

Hee Seok Song*

Software change is an essential operation for software evolution. To maintain the system competently, managers as well as developers must be able to understand the structure of the system but the structure of software is hidden to the developers and managers who need to change it. In this paper, we present a system (OntCIA) for supporting change impact analysis for rating and billing domain based on the semantic web technology. The basic idea of OntCIA is to build a domain knowledge base using an OWL ontology and RDF to implement change impact analysis system that would support the managers and software developers in finding out information about structure of large software system. OntCIA allows users to incrementally build an ontology in rating and billing domain and provides useful information in response to user queries concerning the code, such as, for example "Find the modules which have a role for confirming new subscription". The strengths of OntCIA are its architecture for easy maintenance as well as semantic indexing by automatic reasoning.

Key words : Semantic Web, Ontology, Change Impact Analysis

* Hannam university, Department of Business Administration