능동문서: 서식설계자의 프로그램

남 철 기[†] · 배 재 학[††] · 유 해 영[†††]

요 약

웹은 정보 제공원으로서 중요한 역할을 하며 대부분의 웹 응용프로그램은 문서 중심이다. 문서는 문서설계자의 의도를 함축하고 있으며 이는 업무처리 과정의 자동화에 적극적으로 활용될 수 있다. 이러한 문서기능의 본질 파악을 통해 본 논문에서는 특별한 경우, 문서를 실행 가능한 컴퓨터 프로그램으로 보는 시각으로 접근하였다. 이를 위해 서식, 지식베이스, 규칙 그리고, 질의로 구성되는 능동문서 모델을 제안하였다. 이 모델의 각 요소는 문서의 재사용과 상호 운용성을 위해 XML로 일관되게 표현된다. 소개한 능동문서는 사용자 인터페이스를 제공하는 수동적인 역할 뿐만 아니라 문서설계자가 의도하는 문서처리 절차와 업무규칙을 기계가 읽고 추론하여 처리할 수 있게 하는 문서이다. 이를 통해 문서와 기계가 상호작용을 할 수 있으며 다른 응용 프로그램과 협력할 수도 있다. 이러한 능동문서의 적용 가능함을 보이기 위해 기업간 거래(B2B) 시스템에서 구매주문 처리의 예를 보였다. 서식문서를 컴퓨터 프로그램의 시각으로 바라보는 본 논문의 접근법을 통해 본 연구는 문서중심의 지능적인 응용프로그램 개발을 가속화하는 발판을 마련할 수 있을 것이다. 요컨대 본 논문에서 제시한 능동문서는 지식표현 및 처리기능이 내장되어 있는 바, 시맨틱 웹(Semantic Web)이 추구하는 문서의 역할을 담당할 수 있을 것으로 기대한다.

Active Documents: Programs by Form Designers

Chul-Ki Nam[†] · Jae-Hak J. Bae[††] · Hae-Young Yoo[†††]

ABSTRACT

The Web plays an important role as information source and most Web applications are document-centric. A document implies an intention of its own designer, which can be utilized actively in automation of business processes. Through an understanding of an intrinsic nature of a document function, we can see a document as an executable computer program in a special case. For this approach, we propose an active document model that is composed of form, knowledge base, rules, and queries. For reusability and interoperability of a document, each component of the proposed model is uniformly represented in XML. The proposed active document not only plays a passive role in providing user interfaces, but also is a document that a machine can infer and process with reading a procedure of document processing and business rules intended by document designers. Through this approach, document can interact with machines and can cooperate with other applications. For applicability of our active document, we show a case study for the processing of purchase orders in a B2B e-Commerce system. This paper is expected to provide the framework of accelerating the development of intelligent applications through our approach regards form document as a computer program. In short, the proposed active document contains knowledge representation and processing method, consequently our document will play an important role in providing a concept of document of pursuing in Semantic Web.

키워드 : 능동문서(Active Document), 컴퓨터 프로그램(Computer Program), XML, 논리 프로그래밍(Logic Programming), 규칙 마크업 언어(Rule Markup Language)

1. 연구 동기

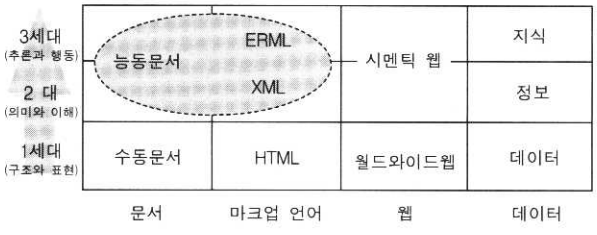
월드와이드웹(이하 웹)은 1990년대 정보통신 발전에 견인차 역할을 하였다. 이러한 웹에 기반한 응용프로그램은 대부분 문서중심(Document-Centric)이며 컴퓨팅 패러다임도 기존의 처리 및 객체 지향 컴퓨팅에서 문서지향 컴퓨팅으로 이행하고 있다[1]. 문서중심의 웹 응용프로그램에서 웹 문서의 중요성은 더욱 더 부각되고 있다. 현재 웹 문서를 표현하기 위해 주로 사용되는 언어는 HTML(Hypertext Markup Language)이다. HTML로 표현된 문서는 내용중심의 정보표현보다는 인간이 보기 편하도록 외형을 제시하는데 중점을 두었다. 이러한 이유로 에이전트가 처리할 문서의 내용은 단지 데이터에 불과하다. 만약 데이터 사이에 연관성을 표현할 수 있는 구조가 더해진다면 데이터는 정보로서의 의미를 갖게 된다. 정보는 에이전트의 추론을 통해 새로운 정보를 만들어 내게 되는데, 정보들이 체계화되어서 의사결정이나 행동에 영향을 줄 수 있다면 이것은 지식이라고 할 수 있다. 이를 통해 지능적인 웹 응용프로그램의 구현이 가능하다[2].

차세대 웹은 기계가 인간을 대신하여 웹 상의 정보를 추출, 통합, 가공, 그리고 추론하는 것을 목표로 한다. 이러한

† 준 회 원 : 한국오라클(주) 기술서비스본부
†† 종신회원 : 울산대학교 컴퓨터정보통신공학부 교수
††† 정 회 원 : 단국대학교 정보컴퓨터학부 교수
논문접수 : 2003년 4월 26일, 심사완료 : 2003년 9월 8일

취지로 웹에 대한 새로운 시각으로 다양한 연구[3]가 현재 진행중이다. 또한, 문서의 의미적 내용을 기계가 이해하지 못하는 수준을 전통적 의미의 웹이라고 한다면, 정보와 지식을 처리할 수 있는 웹을 시맨틱 웹(semantic web)[4]이라 한다. 시맨틱 웹은 웹 상의 정보를 기계가 쉽게 읽고 이해할 수 있도록 잘 정의된 의미를 부여하여 인간과 기계간의 협동작업을 원활하게 하기 위해서 제안되었다. 위와 같이 현재 웹의 사용자가 더 이상 인간에게 한정되지 않고, 다양한 소프트웨어 에이전트로 확대되고 있으며, 이러한 사실로부터 인간과 에이전트간에 웹 문서의 지식공유가 반드시 필요하다는 것을 알 수 있다. 이를 위해 등장한 마크업 언어가 XML이다. XML은 태그가 고정되어 있지 않고 확장될 수 있으며, 여러 분야에서 XML을 기반으로 응용분야 나름대로의 데이터 및 문서교환을 위한 표준화 작업의 추진을 가속화 시켰다[5].

한편, 본 논문에서는 시맨틱 웹의 목표에 부합하는 문서를 제공하기 위해 새로운 시각으로 문서를 조명하였다. 즉, 서식문서를 단순한 사용자 인터페이스를 제공하는 수동적인 문서로만 사용하는 기존의 시각을 달리하고 이를 컴퓨터 프로그램으로 보는 관점에서 접근하고자 한다. 일반적으로 처리의 대상이 되는 문서는 수동적인 것이 아니고, 그 문서를 설계한 사람의 의도인 지식을 함축하고 있어서 능동적이다[6]. 한 예로 HTML 기반의 서식문서는 사용자 인터페이스를 제공하는 역할을 할 뿐, 문서서식 설계자가 지향하는 업무처리 절차나 로직을 내포하지는 않는다. 그러나 서식문서에는 그것에 대한 처리방법이 함축되어 있고, 이렇게 내재된 절차적 지식을 업무처리 과정의 자동화에 적극적으로 활용하여 지능적인 웹 어플리케이션을 구현할 수 있다. (그림 1)은 지금가지 기술한 각 분야별 발전단계를 나타낸다. 이 중에서 능동문서(Active Document)와 시맨틱 웹은 그 추구하는 바가 유사하다. 웹 문서에 인간뿐만 아니라 기계도 읽고 이해할 수 있는 방법으로 지식을 표현함으로써 지능화되고 자동화된 서비스를 제공하자는 것이 주된 목적이다.

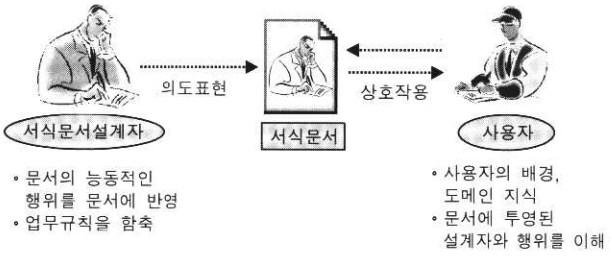


(그림 1) 분야별 발전단계

2. 연구개요 및 기여도

본 논문에서는 문서에 대한 일반적인 시각을 달리하여 문서기능의 본질을 파악하고 이를 통해 지능적인 응용프로그램의 구현 가능성을 도출해 내고자 한다. 문서는 행위자체에 초점을 둔 기능적인 관점과, 문서의 장(Chapter), 절(Section), 문단(Paragraph) 등과 같은 외형에 주안점을 둔 물리적인 관점으로 나누어 볼 수 있다[7]. 문서의 기능적인 관점에서 볼 때, 문서는 수동문서(Passive Document)와 능동문서로 대별된다[1]. 수동문서는 내용과 구조화된 표현만을 가지고 있는 문서이다. 능동문서는 내용과 구조화된 표현뿐만 아니라, 문서의 동적인 특성을 포함하는 문서이다[8].

본 논문에서는 인지 과학적인 측면에서 문서의 능동성을 파악하고, 이를 통해 서식문서를 능동적으로 실행 가능한 컴퓨터 프로그램으로 보는 관점으로 접근하였다. 즉, 일반적으로 자료처리의 대상이 되는 문서는 수동적인 것이 아니고, 그 문서의 설계자가 문서 작성자와 상호작용을 할 때 필요한 지식을 함축하고 있다는 점에서 능동적이라고 본다. 여러 종류의 문서 중에서 서식문서를 대상으로 하였다. 그 이유는 서식문서의 경우 (그림 2)에서와 같이, ① 사람이 단순히 문자를 해독하여 그 문서를 작성하는 것이 아니고, ② 주어진 서식에 함축되어 있는 바, 서식설계자가 지향하는 업무처리과정 및 업무규칙을 추론하여, ③ 그의 의도를 능동적으로 재구성해 가는 이해과정이 필요하다는 것이 잘 드러나는 문서이기 때문이다.



(그림 2) 서식문서 작성의 본질

(그림 2)에서 서식문서는 서식설계자의 의도인 지식을 함축하고 있으며 이러한 지식을 처리하기 위해서는 기계가 지식을 읽고 추론할 수 있어야 한다. 이를 위해서는 서식설계자의 의도인 ① 문서작성규칙, 업무규칙과 같은 명시적 지식과, ② 문서처리 절차와 같이 문서에 내재된 암묵적 지식을 용이하게 표현할 수 있어야 한다. 또한, 이러한 지식은 구조적인 양식으로의 변환을 통해 교환함으로써 궁극적으로 다양한 응용 시스템에서 활용할 수 있는 언어로 설계되어야 한다. 이에 본 논문에서는 XML 형식의 규칙 마크업 언어를 고안하였다.

제안하고자 하는 능동문서는 그 내부에 선언적 지식을 포함하며 추론엔진을 통해 문서제어와 처리에 대한 자동화를 지원하는 컴퓨터 프로그램이다. 이를 위해 문서의 모델을 재정립하였다. 제시하는 능동문서 모델은 서식, 지식베이스, 규칙 그리고, 질의로 구성된다. 모델의 각 요소는 문

서의 재사용과 상호 운영을 위해 XML로 일관되게 표현된다. 이 중에서 규칙은 Prolog 규칙의 XML 표현이며 이를 ERML(Executable Rule Markup Language)[9]이라 칭한다. ERML은 XML의 장점[10]을 그대로 가지고 있을 뿐만 아니라, 규칙의 교환과 저장도 가능한 마크업 언어이다. 본 논문의 능동문서는 동적인 특성을 가지고 있을 뿐만 아니라, 응용프로그램 인터페이스와 유사한 상호 작용하는 요소를 프로그램 형태로 포함하고 있다. 일반적으로 프로그램과 문서는 서로 다른 객체로 취급되어져 왔지만, 제안하는 능동문서는 이러한 이질적인 두 가지 특성을 함께 내포하는 문서이다.

문서를 실행 가능한 컴퓨터 프로그램으로 보는 본 논문의 접근법을 통해 기대할 수 있는 점은 다음과 같다. ① 지금까지 문서의 역할은 단순히 정보의 표현 및 저장, 그리고 교환에 국한되었었다. 여기에서 더 나아가 문서에 내포되어 있는 문서설계자의 지식을 식별하여 다양한 지식의 사용자가 공유할 수 있도록 하는 것은 문서의 진화를 위한 새로운 발판을 마련할 것이라 기대한다. 또한, ② 서식설계자의 의도를 명시적으로 문서에 포함시켜 이를 업무처리 과정의 자동화에 적극적으로 활용함으로써 문서처리의 효율성을 기대할 수 있다. ③ 능동문서에 대한 본 논문의 접근법은 문서와 프로그램에 대한 시각을 통일(Unification)할 수 있으며, 또한 문서처리와 프로그램실행, 그리고 질의처리에 대한 시각을 단일화할 수 있다. 마지막으로 ④ 능동문서 모델을 XML로 일관되게 표현하였기 때문에 이상적인 전자문서의 요건을 갖추고 있는 XML 문서의 장점을 그대로 가질 수 있다. 요컨대 능동문서는 문서중심의 응용프로그램 개발을 가속화하는 토대를 마련할 수 있을 것으로 기대한다.

3. 관련 연구

3.1 규칙 마크업 언어(Rule Markup Language)

규칙은 컴파일러 기술, 데이터베이스, 논리 프로그래밍, 그리고 인공지능 등 다양한 분야에서 사용되어져 왔다. 문서에 함축되어 있는 지식은 문서에 기반 한 다양한 응용 시스템에서 중요한 역할을 한다[11]. 지식을 규칙으로 표현하여 지식을 공유하는 데에는 다음과 같은 기술적인 제약이 있다. ① 규칙을 표현하는 방법이 사용하는 언어에 따라 이질적이다. ② 규칙을 추론하는 도구가 다양하여 단일화된 표준이 없다. 이러한 문제를 해결하기 위해 등장한 공동체가 RuleML[12]이다. <표 1>에 기술된 바와 같이 현재 RuleML 공동체의 주 관심사는 규칙이나 케이스 등과 같은 지식표현 수단을 사용하는 것이다. 그리고 이것을 자료 및 문서교환의 표준이며 종속되지 않은 XML로 표현한 후 이들을 교환하고자 하는 것이다.

3.2 능동문서에 대한 기존 시각

. 능동문서는 소프트웨어 컴포넌트처럼 다루어질 수 있으며 시스템의 구성요소가 될 수도 있는 문서이다. 인터넷 기반의 많은 응용이 문서 중심의 컴퓨팅 모델로 전환되고 있으며 능동문서 또는 문서 에이전트[14] 등의 형태로 최근 활발히 연구되고 있다. 이들 연구는 문서를 내용을 포함하는 수동적인 역할 뿐 아니라 행위를 정의하고 다른 문서와의 상호 관계를 통해 다른 응용 어플리케이션과 협력할 수도 있다는 것에 초점을 두고 있다. 행위는 자체적으로 정의된 액션일 수도 있고 사용자의 액션에 대한 반응으로 정의될 수도 있다. 또한 문서 자신을 네트워크를 통해 이동할 수 있는 기능이 포함되기도 한다. 행위에 랜더링과 같은 수동적인 행위만을 포함한 문서를 수동문서라 하고 다른 어플리케이션에 의해 행위가 일어나게 되는 것을 반응문서, 그리고 문서가 스스로 액션을 일으킬 수 있는 경우를 능동문서 또는 문서 에이전트라고 분류하기도 하였다. 특히 XML 기술의 확산으로 쉽게 문서를 처리하고 데이터를 이동하는 것이 가능하게 되어 문서의 능동문서화가 빠르게 진행되고 있다.

능동문서라는 용어는 1988년에 Spinrad[15]에 의해 처음 언

〈표 1〉 규칙 마크업 언어(RuleMLs)의 유형[13]

RuleML 명칭	설 명
Case Based Markup Language (CBML)	분산환경에서의 사례 기반 추론을 위한 RuleML로서, 케이스의 재사용성과 이질적인 시스템간의 상호 운용성 보장을 위한 XML 기반의 케이스 표현언어
Business Rules Markup Language (BRML)	이질적인 규칙 기반 시스템들간에 규칙을 교환하기 위해 공통의 규칙 구조 명시
Agent-Object-Relationship Markup Language (AORML)	소프트웨어 에이전트가 업무처리, 상호작용 처리, 사건의 순서, 행위, 활동 및 통제 등을 처리할 수 있도록 업무규칙을 XML로 표현
Artificial Intelligence Markup Language (AIML)	단순 패턴매칭 기법을 적용한 채팅 로봇인 ALICE (Artificial Linguistic Internet Computer Entity)를 위한 XML 명세
Universal Rule Markup Language (URML)	이질적인 인공지능 응용체계들이 상호 입력자료와 출력자료를 공유하기 위해 입출력 자료를 XML로 표현
Relational-Functional Markup Language (RFML)	call-by-value 표현을 사용하는 로직 프로그래밍언어인 Relfun의 XML 버전

〈표 2〉 능동문서 시스템의 비교

시스템	Active Tioga : Edit Notifications[16]	Scripted Documents[17]	Interleaf[18]	Quill[19]	ActiveForm (본 논문)
행위 (Activity)	일반적인 계산	일반적인 행위 (action)의 순서	일반적인 계산	SGML의 복잡한 의미 루틴	• 일반적인 계산 • 데이터 검증 • 업무규칙에 따른 문서처리 • WfMS의 워크플로우 수행
트리거링 (Triggering)	사용자 행위 ; 문서 열기, 스크롤, 편집	명시적으로 스크립트 동작을 시작할 때	출력, 편집, 선택, 메뉴를 클릭할 때	출력, 공유데이터 수정할 때	• 실행요구 발생시
코드 위치 (Code location)	별도의 등록된 프로시저에 저장됨	스크립트는 별도의 데이터베이스에 저장	문서	디자인 문서 (문서 스타일)	• 별도의 외부 XML 파일에 저장되며 능동문서에서 참조 되어 포함됨
문서모델 (Document model)	• 노드 구조 WYSIWYG • 풍부한 확장성	• 베타-문서 편집기	• 계층적, 확장가능한 WYSIWYG	• 계층적 SGML WYSIWYG	• 계층적 XML WYSIWYG • 문서는 구조, 표현, 지식베이스, 규칙, 질의를 나타내는 각각의 XML 파일로 구성됨

급되었다. 그 후 여러 연구에서 능동문서의 개념을 사용하였다[16, 17]. 하지만, 능동문서의 응용 영역에 따라 서로 다른 개념을 사용한다. 접근방법에 의해 나누어 볼 때 클라이언트 영역에서 능동문서 처리를 하는 연구와 서버 영역에서 처리하는 연구로 나누어 볼 수 있다. 또한, 전자출판, WfMS, 그리고 메일 시스템에 능동문서의 개념을 도입하기도 하였다. 능동문서 시스템은 문서를 중심으로 사용자 상호작용이나 통신 등을 지원하는 시스템이다. 이는 내용과 행위정의를 내포하는 행동 가능한 문서를 통해 지능적인 어플리케이션을 구현하고자 하는 접근 방법이다. <표 2>에서는 Terry와 Bakers가 제시한 능동문서의 주요문제[18]에 근거해서, 본 논문의 원형 시스템인 *ActiveForm*을 대표적인 능동문서 시스템과 비교하였다.

<표 2>에서 본 논문의 원형 시스템을 제외한 각 시스템은 문서 내의 행위를 명세하기 위해 고유의 포맷을 사용한다. 이것은 능동문서에 포함되어 있는 행위가 이종의 시스템과 교환되는데 걸림돌이 된다. 이러한 문제를 해결하기 위해 본 논문에서는 XML로 행위를 기술하여서, 이종의 시스템간에도 행위 교환이 용이해진다. 능동문서 시스템의 공통된 목표는 일상적으로 사용하는 문서라는 객체를 사용자에게 제공함으로써 사용하기에 쉬우면서도 강력한 어플리케이션을 제공하자는 것이다.

4. 능동문서에 대한 새로운 접근법

4.1 서식설계자의 의도가 내장된 능동문서

본 논문에서 제안하는 능동문서는 외형상으로는 일반적인 형태의 서식문서다. 서식문서는 특정한 업무목적을 위해 만들어진 문서이다. 수많은 유형의 서식들이 존재하는데, 이러한 문서들은 대개의 경우 사용자 인터페이스만 제공하는 역할을 하고 있다. 일상적으로 사용하고 있는 업무용 서식문서에는 문서 설계자가 문서에 반영하고자 하는 의도가 함축되어 있다. 이러한 의도는 ① 문서작성규칙, 업무규칙과 그리고 문서처리 절차와 같은 명시적 지식과, ② 서식설계자의 배경지식과 같은 암묵적 지식이다. 이러한 지식은 문서에 명시적으로 표현되어 있지 않을 수도 있다. 따라서 업무용 서식문서를 작성하기 위해서는 문자 언어로 기록한 텍스트 속에 함축된 서식설계자의 의도를 이해하고 서식문서 사용자가 가지고 있는 기존의 지식과 새로 얻은 지식을 통합함으로써 새로운 의미 체계를 재구성하는 과정이 필요하다. 또한, 서식문서를 서식설계자의 의도대로 작성하기 위해서는 주어진 서식에 함축되어 있는 바, 서식설계자가 지향하는 업무처리과정 및 업무규칙을 추론하여 그의 의도를 능동적으로 재구성해 가는 이해과정이 필요하다.

위와 같이 서식설계자의 의도를 내장한 문서에 기반하여 지능적인 문서처리를 하기 위해서는 효과적인 지식표현방법이 필요하다. 요컨대 서식설계자의 의도를 명시적으로 문서에 포함시켜서 이를 업무처리 과정의 자동화에 적극적으로 활용한다면 문서처리의 효율성을 기대할 수 있다[20]. 다음 절에서는 지식표현과 이를 위한 언어의 요건 및 Prolog 언어에 대해 기술한다.

4.2 서식설계자의 의도표현

주지하는 바와 같이 지식 표현이란 실세계의 지식을 기계와 사람이 동시에 이해할 수 있는 형태로 표현하는 것을 의미한다. 이는 목적달성에 부합되는 구조를 가져야 할 뿐만 아니라 추론의 효율성, 지식 획득의 용이성, 저장의 간결성 및 표현의 정확성, 그리고 다양성 등을 갖추어야 한다. 웹의 창시와 발전을 주도한 팀 버너즈-리는 시맨틱 웹을 위한 언어의 요건으로 다음과 같이 다섯 가지 특성을 언급한 바 있다[21]. ① 간결한 문법(Compact Syntax), ② 잘 정의된(Well-defined) 시맨틱, ③ 지식을 표현하기에 충분한 표현력, ④ 효과적이고 강력하며 기계가 이해하여 처리 할 수 있는 추론 메커니즘을 가져야 하며, 그리고 ⑤ 대용량의 지식베이스를 구축할 수 있어야 한다는 것이다. 이 중에서 특성 ③과 ④는 서로 상반되는 특성이므로 동시에

만족시키기가 어렵다.

위의 다섯 가지 특성은 단지 지식표현과 계산(Computation)에 주안점을 두고 있다. 지식의 교환 및 공유를 위한 특성은 빠져있다. 따라서 다음과 같은 특성이 추가 되어야 한다. ⑥ 지식은 이종의 여러 시스템에서 교환 및 공유가 가능한 표현이어야 한다. 위 ①, ②, ③, ④의 특성을 만족시키기 위해 본 논문에서는 선언적 지식표현을 하는 Prolog를 사용하였다. 그리고 ⑤의 특성을 지원하기 위해 객체관계형 데이터베이스를 이용하였다. 마지막으로 ⑥의 특성을 위해 Prolog로 표현되어 있는 규칙을 변환기를 통해 XML 형식의 규칙인 ERML로 변환하였다. ERML에 관한 내용은 5장에서 자세히 기술된다.

일반적으로 XML 기반의 지식표현은 기계가 쉽게 읽고 이해할 수 있도록 의도하여 만들어진 언어이기 때문에 인간이 읽고 이해하기가 상대적으로 어렵다. 이에 본 논문에서는 하나의 지식에 대해서 Prolog와 XML 표현을 갖게 된다. 이런 경우 표현된 지식에 대해 일관성문제가 대두된다. 이러한 문제를 해결하기 위해 본 논문에서는 데이터베이스 트리거를 이용해 Prolog 규칙과 ERML과의 일관성유지가 가능하게 하였다. 즉, 객체관계형 데이터베이스에 저장되어 있는 Prolog 규칙이 변경되는 경우에 update 트리거가 실행되어 대응하는 ERML도 변경된다. 한편, 웹 문서에 지식을 포함시키기 위해 (그림 3)과 같이 Conceptual Graph(CG)를 이용한 연구[22]도 있다. (그림 3)은 "John believes that Mary has a cousin who has the same age as her."라는 문장을 CG, KIF(Knowledge Interchange Format), RDF(Resource Description Framework)와 같은 지식 표현언어로 표현한 것이다.

전술한 바와 같이 (그림 3)에서 RDF로 기술한 지식은 CG와 KIF에 비해 상대적으로 이해하기 어렵다. 한편 (그림 4)에서는 Prolog로 지식을 표현한 후 이를 처리하는 예를 보여주고 있다.

• Prolog

(1) 규칙표현

```
person (john, 33).
person (mary, 20).
person (jane, 20).
cousin (Person1, Person2) : - parent (Parent1, Person1),
parent (Parent2, Person2), sibling (Parent1, Parent2).
believe (Person, Facts) : - person (Person, _), Facts.
```

• CG (Conceptual Graph)

```
<KR language = "CG">
Age < Property ; // Declare Age as a subtype of Property
Cousin(Person, Person) {Relation type Cousin} ;
[Person : "John"] <- (Believer) <- [Descr : [Person : "Mary"] - { (Chrc) -> [Age : *a] ;
                                                                 (Cousin) -> [Person] -> (Chrc) -> [*a] ;
                                                               } ] ;
</KR>
```

• KIF (Knowledge Interchange Format)

```
<KR language = "KIF">
(Define-Ontology Example (Slot-Constraint-Sugar topLevelOntology))
(Define-Class Age (?X) : Def (Property ?X))
(Define-Relation Cousin(?s ?p) : Def (And (Person ?s) (Person ?p)))
(Exists ((?j Person))
     (And (Name ?j John) (Believer ?j  (Exists ((?m Person) (?p Person) (?a Age))
                                          (And (Name ?m Mary) (Chrc ?m ?a)
                                               (Cousin ?m ?p) (Chrc ?p ?a)
                                          ))
                               )))</KR>
```

• RDF (Resource Description Framework)

```
<!-- RDF notation (with allowed abbreviations) ; this file is named "example" -->
<RDF xmlns : rdf = "http://www.w3.org/TR/WD-rdf-syntax #" xmlns : t = "http://www.bar.com/topLevelOntology" >
   <Class ID = "Age" > <subClassOf resource = "t #Property"/ > </Class >
   <PropertyType ID = "Cousin" > <comment > Relation type Chrc (Characteristic) </comment >
                                 <range resource = "t #Person"/ >
                                 <domain resource = "t #Person"/ > </PropertyType > </RDF >

<RDF xmlns = "http://www.w3.org/TR/WD-rdf-syntax #" xmlns : x = "http://www.bar.com/example"
             xmlns : t = "http://www.bar.com/topLevelOntology" >
 <Description aboutEach = "#Statement_01" > <t #Believer > John </t #Believer > </Description >
 <t #Person bagID = "Statement_01" > <t #Name > Mary </t #Name >
                                    <x #Chrc > <x #Age ID = "age" > </x #Age > </x #Chrc >
                                    <x #Cousin > <t #Person > <x #Chrc resource = "#age"/ > </t #Cousin >
</t #Person > </RDF >
```

(그림 3) 지식표현의 비교

```
(2) 질의 (mary와 jane이 사촌관계일 때)

: - believe (john, (cousin (mary, jane), (person (mary, AgeOfMary),
    person (jane, AgeOfJane), AgeOfMary =:= AgeOfJane))).

(3) 질의 결과

Yes
```

(그림 4) Prolog로 지식표현 및 처리를 한 예

(그림 4)에서와 같이 Prolog로 지식표현을 용이하게 할 수 있으며, 표현된 지식은 범용적으로 널리 사용되는 Prolog 추론엔진을 통해 처리가 가능하다는 점이 CG, KIF, RDF와 다른 점이다. 서식 설계자의 의도는 이종의 지식의 교환과 공유를 위해서 XML 형식으로 표현 가능해야 한다. 이에 본 논문에서는 Prolog 규칙을 XML로 변환하는 방안을 모색하였다. 이를 가능하게 하는 것이 본 논문에서 제안하고 있는 실행가능 규칙 마크업 언어이다.

5. 실행가능 규칙 마크업 언어(Executable Rule Markup Language)

본 논문에서는 문서에 문서설계자의 의도 즉, 지식을 표현하기 위해 절차적인 언어를 사용하지 않고 선언적인 지식표현이 가능한 Prolog 언어를 사용하였다. Prolog로 문서에 내재된 업무로직을 규칙의 형태로 식별하고 정의함으로써 용이하게 의사소통을 할 수 있고, 이해하기가 쉬우며, 그리고 응용프로그램 로직과 분리하여 규칙의 관리가 가능할 수 있다. 또한, Prolog는 선언적인 지식표현을 지원하며 실행 가능한 명세(Executable Specifications)를 기술할 수 있는 프로그래밍 언어이다. 그리고, 표현된 지식을 추론할 수 있는 추론엔진도 내장하고 있다. 따라서, Prolog 프로그램의 XML 표현인 ERML[9]은 실행 가능한 규칙 마크업 언어라고 할 수 있다.

ERML은 Prolog 규칙의 XML 표현이며 서식설계자의 의도를 소프트웨어 에이전트가 읽고 이해할 수 있는 형태의 언어이다. 또한, 능동문서 처리 프레임워크[23] 기반 하에서 실행 가능한 규칙 표시 언어이다. 이로써 지식의 상호 운용성을 제공할 수 있는 기반을 마련할 수 있다. 광의의 의미에서 보면 XML을 기반으로 규칙을 표현하고자 한다는 점에서 ERML도 규칙 마크업 언어의 한 시도로 볼 수 있다. 그러나 ERML은 단순히 규칙을 표현하는 데 그치는 것이 아니다. 즉, 서식문서에 내포되어 있는 서식설계자의 지식을 식별하여 명시적으로 표현하고 이를 업무처리과정에 적극적으로 활용한다. 또한, ERML은 Prolog 추론엔진에서 실행가능하고 또한, 워크플로우 관리시스템과 연동되어 처리될 수 있다는 점에서 기존의 연구들과는 다르다. 본 논문에서는 Prolog와 XML과의 관계[24]를 파악함으로써 ERML을 고안하였다. Prolog와 XML과의 상호 관계를 살펴보면 XML 문서의 논리적 구조 자체는 구조적인 트리(Tree)로 나타낼 수 있다. 또한, 이러한 트리는 Prolog의 구조체 객체로 사상될 수 있다[4].

XML 문서는 요소(Element)로 구성되어 있고 각각의 요소는 <태그> … </태그>의 형식이다. Prolog 규칙을 XML 형식으로 변환하기 위해서는 Prolog에서 사용하는 상수(Constants), 변수(Variables), 구조체(Structures)와 같은 Herbrand Term과 사실(Fact), 규칙(Rule)과 같은 Horn Clause를 XML 요소로 변환하기 위한 명명규칙이 필요하며 이는 <표 3>과 같다.

〈표 3〉 Prolog-XML 요소간의 명명규칙

Herbrand Term	Element	Horn Clause	Element
원 자	*<atom>*	술 어	*<relator>*
숫 자	*<number>*	관계기호	*<relator>*
논리변수	*<variable>*	사 실	*<hn><relationship>*
구 조 체	*<structure>*	규 칙	*<hn><relationship>*

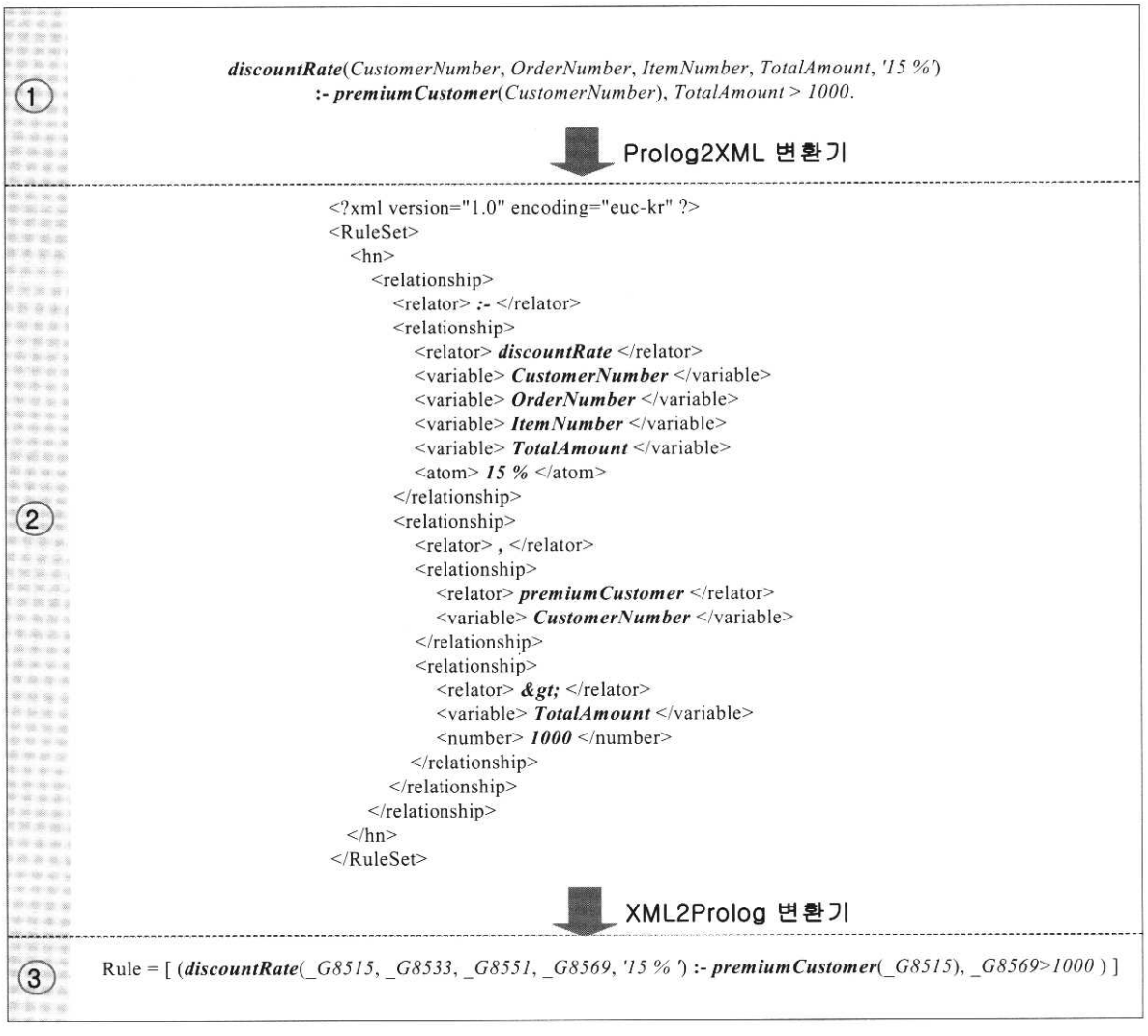
(그림 5)는 <표 3>의 명명규칙에 근거해서 작성된 ERML의 DTD(Document Type Definition)를 나타낸다.

```
< !ELEMENT   ruleset        ( hn* ) >
< !ELEMENT   hn             ( relationship* ) >
< !ELEMENT   relationship   ( relator, relationship*,
                              (variable|atom|number )*)>
< !ELEMENT   relator        ( #PCDATA ) >
< !ELEMENT   variable       ( #PCDATA ) >
< !ELEMENT   atom           ( #PCDATA ) >
< !ELEMENT   number         ( #PCDATA ) >
```

(그림 5) ERML의 DTD

한 예로 구매관련 B2B 사이트에서 고객에 대한 할인율 정책이 다음과 같다고 하자. "만약 고객이 우수고객이면서 총 구매금액이 1000만원 이상이면 15%를 할인한다." 이와 같이 대부분 IF-THEN 형식의 자연어로 기술된 업무규칙은 다음 (그림 6)의 ①과 같이 Prolog 규칙으로 표현될 수 있다. 이 규칙은 지식의 교환 및 공유를 위해 Prolog2XML 변환기를 통해 ②와 같이 적격한(Well-formed) XML 문서인 ERML로 자동적으로 변환 가능하다. 이 ERML을 이종의 규칙기반 시스템에서 사용하기 위해서는 해당 규칙기반 시스템에 적용 가능한 규칙으로 변환되어져야 한다.

본 논문에서는 실험의 효율을 위해 역방향(Backward) 추론엔진을 가지고 있는 SWI-Prolog[25]를 사용하였다. ERML을 Prolog의 추론엔진에서 수행하기 위해서는 원래의 Prolog 규칙으로 변환하는 작업이 필요하다. 이는 DCGs(Definite Clause Grammars)를 이용하여 만들어진 XML2Prolog 변환기[6]를 통해 변환한다. (그림 6)의 에서는 리스트에 하나의



(그림 6) Prolog 규칙과 ERML의 변환 예

요소로 원래의 할인율 규칙(*discountRate*)이 표현됨을 확인할 수 있다. 이때 각각의 변수는 SWI-Prolog에서 할당한 내부 변수로 표현된다. 리스트의 요소를 표현된 규칙은 질의 처리를 위해 Prolog 지식베이스로 저장된다.

6. 컴퓨터 프로그램으로서의 능동문서 : *activeForm*

본 장에서는 컴퓨터 프로그램의 관점에서 능동문서를 조명하고, 이를 위한 문서모델을 제시한다. 또한, 문서를 중심으로 사용자와의 상호작용을 가능하게 하는 능동문서시스템의 설계목표 및 능동문서의 실행과정을 서술한다.

6.1 응용프로그램을 위한 능동문서 모델

일반적인 의미로 컴퓨터 프로그램이란, 특정한 문제를 해결하기 위해 기계가 사용하는 일련의 지시 및 명령을 말한다. 이러한 컴퓨터 프로그램을 통해 사용자에게 특정한 서비스를 제공하는 응용프로그램을 제공할 수 있다. 응용프로그램은 네 가지 구성요소로 이루어진다. ① 사용자 인터페이스, ② 데이터, ③ 업무 로직, 그리고 ④ 사용자 서비스 요청이다. 전술한 바와 같이 본 논문에서는 서식설계자가 의도하는 지식을 표현하기 위해 Prolog를 사용하였다. 다음 <표 4>는 응용프로그램의 구성요소와 이들이 능동문서에서 어떤 형태로 지원되는지를 나타낸다. 한 예로, 기업간거래(B2B) 응용프로그램은 다음과 같이 구성된다. ① 판매가능한 제품의 목록 또는 서비스, ② 가격 할인정책과 환불규칙과 같은 업무규칙, 그리고 ③ 견적요청과 제품주문과 같은 사용자 질의이다

〈표 4〉 응용프로그램과 능동문서의 비교

응용프로그램	능동문서
사용자 인터페이스	서식(구조, 표현)
데 이 터	지식베이스
업무 로직	규 칙
사용자 서비스 요청	질 의

본 논문에서 제안하는 능동문서는 그 내부에 선언적 지식을 포함하며 문서제어와 처리에 대한 자동화를 지원하는 컴퓨터 프로그램이다. 문서를 프로그램적인 관점으로 접근하기 위해 문서의 모델을 재정립하였다. 문서모델을 구성하는 각각의 요소는 다음과 같이 다섯 가지로 구성된다. ① **구조(Structure)**는 문서의 논리적인 구조이다. 즉, 서식의 각각의 필드집합과 그 각 집합을 구성하는 필드이름, 필드형태와 필드길이를 정의하는 부분이다. ② **표현(Presentation)**은 XML로 표현된 문서의 논리적 구조를 시각적으로 표현하는 XSL 문서이다. 구조를 표현하는 XML 문서에 표현을 나타내는 XSL 문서를 적용하여 HTML 서식문서를 생성한다. 이 서식문서는 사용자 인터페이스의 역할을 한다. ③ **지식베이스(Knowledge Base)**는 Prolog 추론엔진이 추론할 때 필요한 지식베이스를 제공한다. 예를 들면, 판매 가능한 제품의 목록 또는 서비스가 이에 해당된다. ④ **규칙(Rule)**은 업무문서에 함축되어 있는 업무규칙과 문서에 대한 처리방법을 의미하며, 일반적인 계산 로직, 업무규칙과 같이 업무 프로세스를 정의하고 제어하는 규칙, 데이터 검증에 필요한 데이터 무결성의 제약조건, 그리고 워크플로우의 호출 등이 있다. 이러한 규칙은 우선 Prolog 규칙으로 표현된 후에 Prolog 규칙의 XML 표현인 ERML로 변환된다. 그리고 마지막으로 ⑤ **질의(Query)**는 사용자가 서식에 입력한 데이터 검증 및 문서처리를 위한 것이다. 이들 각각의 구성요소는 XML로 일관되게 표현되어 서식문서 내에 함께 포함된다.

6.2 능동문서 시스템의 설계목표

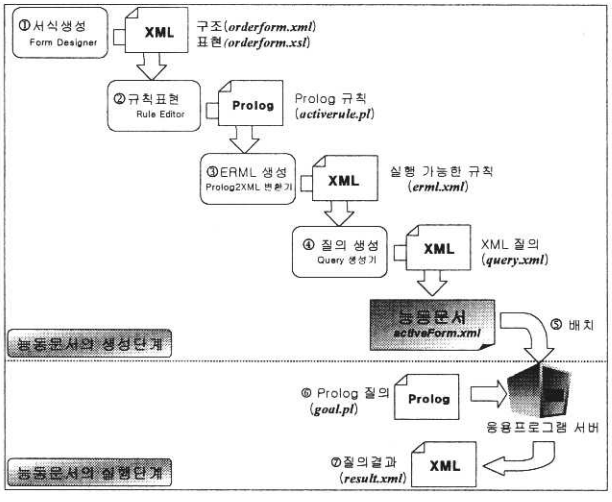
본 논문에서는 다음과 같은 다섯 가지 목표를 염두에 두고 능동문서 시스템을 설계하고자 한다 :

- **일반적(Generic)이어야 한다** : 제안하는 능동문서 시스템은 도메인에 독립적이어야 한다. 즉, 추론엔진은 일반적인 목적으로 널리 사용되는 것이어야 하며, XML 기반의 규칙 언어인 ERML은 특정한 업무 도메인에 제한되지 않아야 한다.
- **통합되어야 한다(Integrated)** : 능동문서를 생성하는 도구는 통합되어야 한다. 그리고 생성된 능동문서는 실행을 위한 환경인 웹 어플리케이션으로 매끄럽게(Seamlessly) 전환되어야 한다.
- **확장 가능해야 한다(Extensible)** : 즉, 능동문서 시스템은 최소한의 작업으로 새로운 업무 로직을 추가할 수 있어야 한다.
- **사용하기 쉬워야 한다(Easy to use)** : 이는 서식설계자가 능동문서를 생성할 때 직관적인 사용자 인터페이스를 제공하는 편집기를 사용할 수 있어야 한다는 것이다.
- **이동이 쉬워야 한다(Portable)** : 능동문서 시스템에 관련된 도구와 구성 요소들이 플랫폼에 종속되지 않아야 한다는 것이다.

이것을 가능하게 하기 위해 자바와 웹 기반의 접근법이 필요하다. 이를 위해 능동문서 디자이너를 설계 및 구현하였다. 이 능동문서 디자이너를 사용하여 다음 절에서 서술하는 능동문서를 생성하게 된다.

6.3 능동문서의 생성 및 실행

능동문서 디자이너로 생성한 능동문서가 추론엔진에서 데이터 검증을 통해 WfMS와 연동되어 처리되는 과정은 다음 (그림 7)과 같다. 이 중에서 ③ 단계인 ERML 생성과 ④ 단계인 질의생성은 규칙편집기로 작성된 규칙에 근거하여 자동적으로 행해진다.

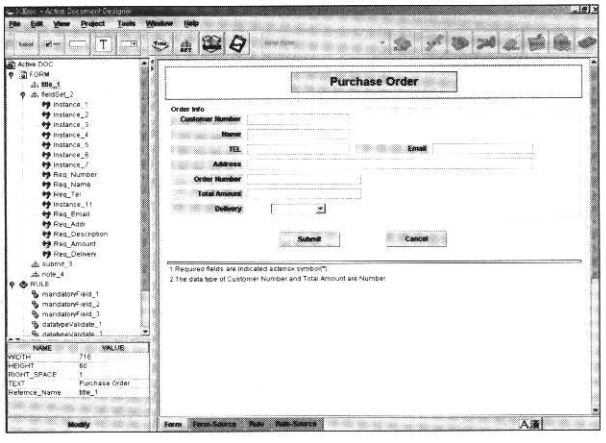


(그림 7) 능동문서의 생성 및 실행과정

6.3.1 능동문서의 생성

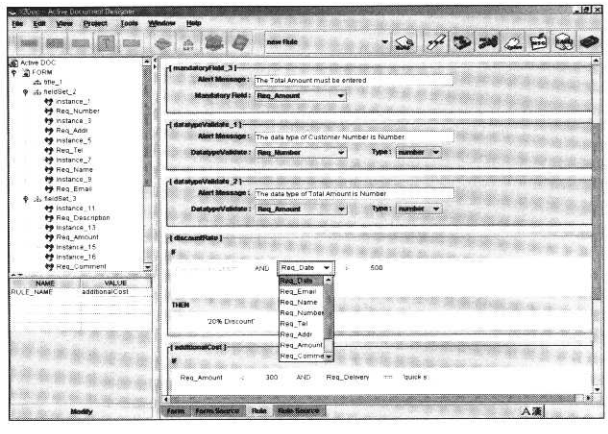
능동문서를 생성하는 단계에서는 다음과 같은 네 가지 작업이 이루어진다. ① 서식생성, ② 규칙표현, ③ ERML 생성 그리고, ④ 질의를 생성한다. 본 논문에서 구현한 서식 디자이너를 이용하여 업무전문가가 WYSIWYG 환경에서 서식문서를 디자인 및 생성한다. (그림 8)의 상단의 아이콘을 사용하여 직관적으로 서식을 디자인 할 수 있다. 서식 디자이너에 사용되는 아이콘으로 문서제목, 문서항목의 집합들, 각 항목의 이름과 형태를 지정할 수 있다. 서식디자이너의 오른쪽 창은 사용자가 서식을 디자인하는 곳으로 문서를 저장하게 되면 XML 형식으로 문서의 구조를 저장하게 된다. 여기에 XSL로 작성된 스타일시트를 적용하게 되면 사용자 디바이스에 맞는 서식문서를 생성하게 된다. 특히, 본 논문에서는 HTML 형식의 서식문서를 생성하게 되며, 서식 디자이너에서 작성된 모습과 거의 유사한 형태의 서식을 웹 브라우저에서 확인할 수 있다. 서식 디자이너 창에서

의 각각의 항목은 서식표현을 위한 높이, 너비, 그리고 항목간 공간정보를 갖게 된다. 또한, 각 항목은 참조이름(Reference Name)을 갖게 되며 이는 규칙편집기에서 사용할 항목변수의 이름이다. 다음은 서식 디자이너에서 서식을 디자인하고 저장한 후 확인한 모습이다.



(그림 8) 구매 요청서식 작성

서식을 디자인 한 다음에는 규칙 편집기를 사용하여 규칙을 작성한다. 규칙편집기에서 작성 가능한 규칙은 ① 문서작성규칙과 같은 무결성규칙, ② 사용자가 입력한 값에 의해 적용되는 값을 추론하기 위한 유도규칙, ③ 사용자가 입력한 값에 의해 특정 이벤트가 발생했을 때 수행해야 할 외부 프로그램 또는 업무 프로세스를 처리하기 위한 반응규칙이다. (그림 9)에서와 같이 규칙편집기의 오른쪽 상단에 있는 아이콘들을 사용하여 직관적으로 규칙을 작성할 수 있다. 지원되는 연산자로는 AND, OR, NOT과 같은 논리연산자와 >, <, >=, <=, ==와 같은 비교연산자가 있다. 규칙편집기로 규칙을 작성 후 저장하게 되면 능동문서 디자이너의 왼쪽에 위치하는 능동문서 프로젝트 탐색기에서 RULE이라는 노드로 생성된다. 이때 표현되는 정보는 규칙의 이름이다. 또한, 규칙이 저장되면 사용자 질의를 위한 질의의 원



(그림 9) 규칙의 편집

형이 자동적으로 생성되게 된다. 이는 프로젝트 탐색기에서 QUERY라는 노드로 생성된다. 규칙편집기를 통해 작성된 규칙은 *activerule.pl* 파일에 저장된다. <표 5>와 <표 6>은 구매요청 서식에 적용 될 규칙들이며, 이는 규칙편집기를 사용하여 직관적으로 작성된다.

〈표 5〉 유도 규칙(판매정책)

규 칙	표 현	
(규칙 1) 할인율	의미	전체 구매액이 1000만원보다 클 경우에는 할인율이 15.0%이다.
	Prolog 규칙	***discountRate***(CustomerNumber, Name, Tel, Email, Address, OrderNumber, TotalAmount, Delivery, '15.0%') :- TotalAmount > 1000.
(규칙 2) 추가비용	의미	제품 배송방법이 Quick Service인 경우, 구매금액에 10%의 추가비용을 더 지불해야 한다.
	Prolog 규칙	***additionalCost***(CustomerNumber, Name, Tel, Email, Address, OrderNumber, TotalAmount, Delivery, '10% Additional Cost') :- Delivery == 'quick service'.

〈표 6〉 무결성 규칙

규 칙	표 현	
(규칙 1) 필수필드	의 미	고객번호, 주문번호, 총 구매금액은 반드시 입력되어야 한다.
	Prolog 규 칙	***mandatoryField_1***(CustomerNumber, Name, Tel, Email, Address, OrderNumber, Total-Amount, Delivery, 'The Customer Number must be entered') :- ***var***(CustomerNumber). ***mandatoryField_2***(CustomerNumber, Name, Tel, Email, Address, OrderNumber, TotalAmount, Delivery, 'The Order Number must be entered') :- ***var***(OrderNumber). ***mandatoryField_3***(CustomerNumber, Name, Tel, Email, Address, OrderNumber, TotalAmount, Delivery, 'The Total Amount must be entered') :- ***var***(TotalAmount).
(규칙 2) 데이터 타 입	의 미	고객번호, 총 구매금액의 데이터타입은 숫자이다.
	Prolog 규 칙	***datatypeValidate_1***(CustomerNumber, Name, Tel, Email, Address, OrderNumber, TotalAmount, Delivery, 'The data type of Customer Number is Number') :- ***not***(***number***(CustomerNumber)). ***datatypeValidate_2***(CustomerNumber, OrderNumber, ISBN, Quantity, TotalAmount, 'The data type of Total Amount is Number') :- ***not***(***number***(TotalAmount)).

질의는 (그림 7)의 ①, ② 번째 단계에서 생성된 서식필드의 데이터 타입과 규칙에 따라 질의 생성기로 자동적으로 생성된다. 능동문서 프로그램을 나타내는 *activeForm.xml* 파일은 ① 추론시에 지식베이스를 제공하는 *KB.xml*, ② 문서의 서식구조와 스타일시트를 포함하는 *orderform.xml*, ③ 문서의 규칙을 나타내는 *erml.xml*, 그리고 ④ 사용자 질의의 원형을 나타내는 *query.xml*을 포함한다. *activeForm.xml*은 다

음과 같이 물리적으로는 4개의 XML 문서로 구성되어져 있지만 논리적으로는 하나의 XML 문서이다.

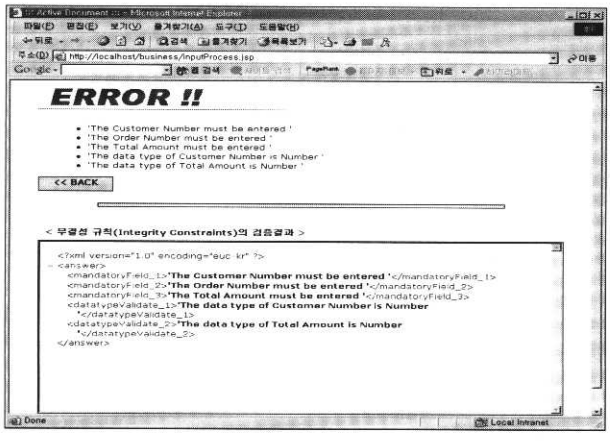
```
<?xml version = "1.0" encoding = "euc-kr"?>
<!DOCTYPE activeForm [
    <!ENTITY knowledgebase SYSTEM "KB.xml">
    <!ENTITY form SYSTEM "orderform.xml">
    <!ENTITY rule SYSTEM "erml.xml">
    <!ENTITY query SYSTEM "query.xml">
]>

<Active_document>
    &knowledgebase;
    &form;
    &rule;
    &query;
</Active_document>
```

6.3.2 능동문서의 실행

능동문서를 실행하는 단계에서는 크게 두 단계를 거친다. **(1 단계)**에서는 ERML 문서를 Prolog 규칙으로 변환하는 단계이다. 이 단계에서는 서식문서의 데이터 검증과 처리를 위해 XML2Prolog 변환기를 사용하여 *erml.xml*로부터 원래의 Prolog 규칙으로 변환하여 Prolog 지식베이스로 저장한다. **(2 단계)**에서는 사용자가 서식에 기입한 데이터에 근거해서 Prolog 질의를 수행한다. 이는 Prolog 추론엔진을 통한 데이터 검증 및 문서처리를 하기 위함이다. 능동문서 생성단계에서 만들어진 능동문서는 어플리케이션 서비스를 위해 웹 어플리케이션 서버에 배치(deploy)된다. (그림 8)에서 서식 디자이너로 작성한 구매 요청서는 웹 브라우저에서 동일한 사용자 인터페이스를 제공하게 된다.

• 무결성 규칙(Integrity Constraints)의 검증

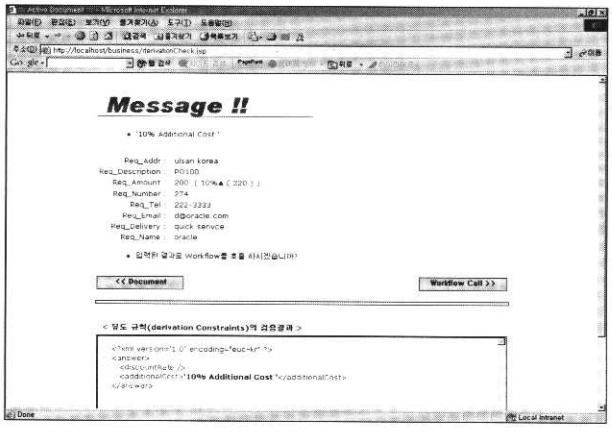
(그림 10)은 구매요청서에 값을 전부 입력하지 않은 경우 발생하는 에러내용을 보여주고 있다. 그리고 결과는 XML 파일(*result.xml*)로 저장된다. *result.xml*의 내용은 다음 그림에서 TEXTAREA에 있는 내용이다.



(그림 10) 무결성 규칙의 검증

• 유도 규칙(Derivation Rule)의 적용

무결성 규칙에 위배됨이 없이 데이터를 기입한 경우에 <표 5>의 유도 규칙이 적용된다. (그림 11)의 경우는 구매자의 총 구매금액이 200만원이고, 배송방식이 Quick Service인 경우에 대한 예이다. 유도 규칙에 의해 배송방법(Delivery)이 quick service인 경우 총 구매 금액에 10% 추가되는 금액을 적용 받는다. 따라서 220만원을 지불해야 한다. 유도규칙이 적용된 후 Workflow Call 버튼을 클릭하게 되면 워크플로우 엔진이 해당하는 워크플로우를 시작하게 된다.



(그림 11) 유도 규칙의 적용

7. 결 론

본 논문에서는 웹 응용프로그램의 사용자 인터페이스로 주로 사용되는 서식문서를 효과적으로 처리하기 위하여 서식문서를 실행 가능한 컴퓨터 프로그램으로 바라보는 시각으로 접근하였다. 제안하고자 하는 능동문서는 외형적으론 서식문서이지만 그 내부에 선언적 지식을 포함하며 추론을 통해 문서제어와 처리에 대한 자동화를 지원하는 컴퓨터 프로그램이다. 이와 같이 문서를 컴퓨터 프로그램으로 보는 시각으로 접근하기 위해 서식, 지식베이스, 규칙 그리고, 질의로 구성되는 문서의 모델을 재정립하였다. 제시한 능동문서 모델의 각 요소를 문서의 재사용과 상호 운용성을 위해 XML로 일관되게 표현하였다. 그 결과 본 논문의 능동문서는 동적인 특성을 가지고 있을 뿐만 아니라, 응용프로그램 인터페이스와 유사한 상호 작용하는 요소를 프로그램 형태로 포함할 수 있었다. 이를 통해 문서와 기계가 상호작용을 할 수 있으며 다른 응용 프로그램과 협력할 수도 있었다.

본 논문에서 제안하는 능동문서를 활용하여 비즈니스 환경에 맞는 응용프로그램을 개발할 때 다음과 같은 장점을 기대할 수 있다. ① 일반적으로 새로운 응용프로그램을 개발할 때 초기 개발 비용 보다는 관리 및 유지보수 비용이 더 많이 든다. 이러한 문제를 해결하기 위한 방법으로 본 논문에서는 응용프로그램의 사용자 인터페이스로 주로 사용되는 서식문서에 업무규칙과 처리 로직을 포함시켜 처리

하였다. 이렇게 함으로써 비즈니스에서 얻어질 수 있는 가치는, 업무에 대한 요구사항의 수정이 필요할 때, 서식문서 설계자가 개발자에게 업무 로직의 수정을 요청하지 않고 바로 업무 로직을 수정/변경함으로써 업무요구사항을 유연하게 만족시킬 수 있다. 또한, ② 선언적으로 정의된 업무 규칙은 프로그래밍 언어로 복잡한 구현 없이 즉시 수행이 가능하므로 급변하는 비즈니스 환경에 유연하게 대처할 수 있다. 마지막으로, ③ 본 논문에서는 구매요청서 처리를 위한 업무 로직을 표현하기 위해 규칙을 사용하였다. 이는 절차적인 언어를 사용하는 일반적인 방법보다 개발 복잡성이 덜 하며 개발 및 유지보수에 드는 시간적인 비용을 절감할 수 있다.

지금까지 문서의 역할은 단순히 정보의 표현 및 저장, 그리고 교환에 국한되었었다. 이러한 역할에서 한 단계 더 나아가 문서에 내포되어 있는 문서설계자의 지식을 식별하여 다양한 지식의 사용자가 공유할 수 있도록 하는 것은, 문서중심의 컴퓨팅 패러다임에서 문서의 진화를 위한 새로운 발판을 마련할 수 있음을 확인하였다. 또한, 능동문서 모델의 요소 중에서 규칙을 표현하기 위한 ERML은 문서에 함축되어 있는 서식설계자의 지식을 표현하기 위한 언어임과 동시에, 웹 공동체가 추구하고 있는 시맨틱 웹을 위한 규칙마크업 언어로서도 그 역할을 할 수 있다. 궁극적으로 본 논문에서 제안한 능동문서는 지식표현 및 그 처리기능이 내장되어 있는 바, 시맨틱 웹에서 시맨틱 웹문서(Semantic Web Document)의 역할을 수행할 수 있을 것이라 기대한다.

참 고 문 헌

[1] C. Paolo, T. Robert and Z. Franco, "Coordination Middleware for XML-centric Applications," 16th ACM symposium on Applied Computing, Madrid(E), Mach, pp.336-343, 2002.

[2] C. -K. Nam, G. S. Jang and J. -H. J Bae, "An XML-based Active Document for Intelligent Web Applications," Expert System with Applications, Vol.25, No.2, pp.165-176, 2003.

[3] SemanticWeb.org, http://www.semanticweb.org/.

[4] W3C Semantic Web, http://www.w3.org/2001/sw/.

[5] 남철기, 김혜경, 배재학, "응용프로그램을 위한 일관된 XML View제공에 관한 연구", 한국정보처리학회 춘계학술대회논문집, 제8권 제1호, pp.1105-1108, 2001.

[6] 남철기, 장길상, 배재학, "능동문서에 대한 새로운 접근법과 그 응용", 한국정보과학회논문지, 제30권 제34호, pp.347-357, 2003.

[7] M. K. Buckland, "What is a document?," Journal of the American Society for Information Science, Vol.48, pp.804-809. 1997.

[8] 남철기, 배재학, "업무규칙을 포함한 능동문서", 한국정보과학회 춘계학술발표대회논문집, 제29권 제1호, pp.352-354, 2002.

[9] C. -K. Nam, G. S. Jang and J. -H. J Bae, "An Active Processing Approach of Web-based Form Documents : Centering around ERML," In Proceedings of The International Conference of Society of Korea Industrial & System Engineering (SKISE), Session A7, A7-2.pdf, 2002.

[10] 남철기, 양재군, 배재학, "검증규칙을 포함한 XML문서", 한국정보과학회 추계학술발표대회논문집, 제28권 제2호, pp. 709-711, 2001.

[11] L. Nonaka et. al., "The concept of Ba : Building a Foundation for Knowledge Creatation," California Management Review, Vol.40, No.3, pp.40-54, 1998.

[12] RuleML Homepage, http://www.dfki.uni-kl.de/ruleml/. 2003.

[13] 이재규, 손미애, 강주영, "확장형 규칙 표식 언어(eXtensible Rule Markup Language) : 설계 원리 및 응용", 한국지능정보시스템학회 춘계학술발표대회논문집, pp.284-293, 2002.

[14] G. Cabri, L. Leonardi and F. Zambonelli, "XML Dataspaces for Mobile Agent Coordination," 15th ACM Symposium on Applied Computing, pp.181-188, 2000.

[15] R. Spinrad, "Dynamic documents," Harvard University Information Technology Quarterly, Vol.Ⅶ, No.1, pp.15-18, 1988.

[16] H. Ahonen et al, "Intelligent Assembly of Structured Documents," Technical Report C-1996-40, University of Helsinki, Department of Computer Science, 1996.

[17] P. Dourish et al, "Extending Document Management Systems with User-Specific Active Properties," Transactions on Information Systems, ACM, Vol.18, No.2, pp.140-170, 2000.

[18] P. M. English and R. Tenneti, "Interleaf active documents," Electronic Publishing, Vol.7, No.2, pp.75-87, 1994.

[19] K. Eckhart and N. Gustaf, "Active Hypertext for Distributed Web Applications," Proc. The 8th IEEE International workshops on Enabling Technologies : Infrastructure for Collaborative Enterprises, pp.297-302, 1999.

[20] 남철기, 배재학, 유해영, "능동문서 : 서식설계자의 의도가 내장된 프로그램", 한국정보처리학회 춘계학술발표대회논문집, 제10권 제1호, pp. 353-356, 2003.

[21] T. Berners-Lee, "The Semantic Web as a Language of Logic," http://www.w3.org/DesignIssues/Logic.html, 2000.

[22] P. Martin and P. Eklund, "Embedding knowledge in Web documents : CGs versus XML-based metadata languages," ICCS '99, pp.230-246, 1999.

[23] C. -K. Nam and J. -H. J. Bae, "A Framework for Processing Active Documents," The 6th Russian-Korean International Symposium On Science and Technology, Proc. KORUS-2002, Vol.1, pp.122-125, 2002.

[24] H. Boley, "Relationships between Logic Programming and XML," Proc. 14th Workshop Logische Programmierung, Würzburg, 2000.

[25] SWI-Prolog's Home, http://www.swi-prolog.org/.

[26] Oracle Workflow, http://otn.oracle.com/products/integration/workflow/workflow_fov.html.

남 철 기

e-mail : cholki.nam@oracle.com

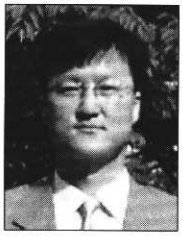
1996년 울산대학교 전자계산학과(학사)

1999년 울산대학교 정보통신대학원 정보통신공학(공학석사)

2003년 울산대학교 대학원 컴퓨터정보통신공학과(공학박사)

1996년~현재 한국오라클(주) 기술서비스본부

관심분야 : 데이터베이스, 지식관리, 시맨틱웹 등



배 재 학

e-mail : jhjbae@ulsan.ac.kr

1981년 중앙대학교 전자계산학과(이학사)

1983년 한국과학기술원 전산과(공학석사)

2003년 포항공과대학교 컴퓨터공학과(공학박사)

1996년~현재 울산대학교 컴퓨터정보통신공학부 정교수

관심분야 : 논리 프로그래밍, 자동 프로그래밍, 지식경영 및 기술, 전략경영 정보시스템, 교육 인적자원 정보시스템 등

유 해 영

e-mail : yoohy@dankook.ac.kr

1979년 단국대학교 수학과(이학사)

1982년 단국대학교 대학원 수학과(이학석사)

1994년 아주대학교 대학원 컴퓨터공학과(공학박사)

1983년~현재 단국대학교 정보컴퓨터학부 컴퓨터과학전공 교수

관심분야 : 소프트웨어공학, 정보화 전략계획수립 및 컨설팅, IT 자원튜닝 등