의미제약 기반의 ebXML BPSS 사례 검증

Validation of ebXML BPSS Instances Based on Semantic Constraints

김형도(Kim Hyoung Do)*, 김종우(Kim Jong Woo)**

초 록

대표적인 전자거래 프레임워크인 ebXML에서 비즈니스 프로세스 명세(BPS: Business Process Specification)는 최종적으로 XML 버전의 BPSS(Business Process Specification Schema)를 준수하는 사례로서 규정되어야 한다. 보다 완전하고 일치되게 XML 버전의 BPSS 사례를 정의하기 위해서는 모든 의미 제약을 검증하는 과정이 필수적이다. 그러나, XML Schema 구조체의 제약으로 인해서 XML버전의 BPSS는 이러한 의미 제약을 완벽하게 규정하고 있지 못하다. 이 논문에서는 최종적으로 실행될 XML 버전의 BPSS 사례에 대한 검증을 지원하기 위해서, BPSS의 XML Schema에 표현되지 못한 의미 제약들을 체계적으로 발견하고, 이들을 명시적으로 표현하여 재활용하는 방법을 제시한다. 이러한 방법으로 XML 버전 BPSS 사례를 편리하게 검증하고, 오류 수정을 안내하며, 기업간 비즈니스 프로세스 표준화와 적용의 효율성을 증대시킬 수 있다.

ABSTRACT

In ebXML, a representative framework for electronic commerce, a BPS (Business Process Specification) should be finally defined as an instance of XML-version BPSS for the configuration of B2B (Business to Business) runtime systems. In order to define the instance more complete and consistent, it is required to validate all the semantic constraints on the instance. Due to the limitations of XML Schema constructs, however, current XML-version BPSS fails to specify formal semantic constraints completely. This paper presents how to find, express and reuse BPSS semantic constraints that could not be explicitly defined in the XML-version BPSS. The method facilitates the validation of XML-version BPSS instances easily with some useful guides for fixing violations of semantic constraints. Furthermore, B2B business processes can be standardized and applied more efficiently and effectively.

키워드 : 비즈니스 프로세스, 비즈니스 프로세스 명세, 의미 제약, BPSS, ebXML

Business Process, Business Process Specification, Semantic Constraints, BPSS, ebXML

* 한양사이버대학교 경영정보학과 교수

** 한양대학교 경영학부 교수, 교신저자

1. 서 론

대표적인 전자거래 프레임워크인 ebXML (Electronic Business eXtensible Markup Language) 의 Business Process Specification Schema (이하 BPSS로 표기) 는 기업간 비즈니스 프로세스 명세 (Business Process Specification, 이하 BPS로 표기) 를 작성하기 위한 스키마를 제공한다. ebXML BPSS는 UML (Unified Modeling Language) [12,14] 버전과 XML 버전 두 가지가 제시되고 있다 [18]. UML 버전의 경우는 단순한 UML 클래스 다이어그램 형태로, ebXML BPSS 내의 모델링 요소들과 그들간의 관계를 표현하고, XML 버전의 개발과 비즈니스 프로세스 명세 과정을 개념적인 수준에서 수행할 수 있도록 지원하기 위한 것이다. XML 버전의 경우는 XML을 기반으로 비즈니스 프로세스를 즉시 실행 가능한 형태로 정의하고 교환할 수 있도록 명세하기 위해 제시되었다.

ebXML BPS의 작성은 크게 3가지 방법 (비즈니스 프로세스 분석 워크쉬트와 가이드라인에 기반한 편집기 도구를 사용하는 방법 [2], UML 버전의 BPSS를 사용하는 방법, XML 버전의 BPSS를 사용하는 방법) 으로 가능하다. 어떠한 방법으로 작성되든지, 궁극적으로 비즈니스 프로세스를 실행하는 측면에서 사용되는 형태인 XML기반의 BPSS 사례가 생성되어야 한다. 이렇게 정의된 BPS는 비즈니스 프로세스에 필요한 업무 문서의 구조를 정의하거나 기업의 기술적인 능력을 정의하는데 참조되며, 실제로 비즈니스 프로세스를 실행하는 과정에서는 비즈니스 프로세스 엔진을 구동하는 시나리오 역할을 수행하게 된다. 따라서 특정한 비즈니스 프로세스와 관련하여 실존하는 의미 제약을 완전하고 일치되게 정의할 수 있는가 하는 것이 BPS를 (재)활용하는데 있어서 매우 중요하다.

완전하고 일치되게 BPS를 정의하기 위해서는 모든 의미 제약을 명시적으로 표현하고 이를 BPS에 적용하여 검증하는 과정이 필요하다. 이러한 제약의 표현과 검증은 3가지 BPS 작성 방법에서 모두 필요하다. 그 중에서도 최종적으로 실행될 XML 버전의 BPS에 대한 검증은 특히 중요하다. XML 버전의 BPSS는 XML Schema로 정의되어 있는데, XML Schema를 이용한 제약의 표현은 매우 제한적이기 때문에 이러한 목적은 달성되기 어렵다. 따라서 XML Schema에서 소화될 수 없는 의미제약은 프로그램으로 구현되든지, 아니면 제약 언어에 의해서 모델링 되어, 검증 받아야 한다. 그런데, XML 버전의 BPSS에서 XML Schema로 표현되기 어려운 제약은 잘 정리되어 있지 않으며, 설명이 된 경우라도 명시적인 방법으로 규정되어 있지 못하기 때문에, 의미 제약을 일치되게 해석하고 검증하기 어렵다. 따라서 XML 버전의 BPS에 대한 검증이 이루어지기 위해서는 먼저 의미 제약을 확인하고 발견하는 작업이 선행되어야 한다. 발견된 의미 제약들을 명시적인 제약 언어를 이용하여 표현하고, 이를 XML 버전의 BPSS 사례에 적용할 수 있다면 보다 완전하고 일치된 BPS를 얻을 수 있을 뿐 아니라, 수정이 필요한 사항을 모델링 과정에서 제시하거나, 일치성 (Conformance) 테스트를 위한 자료 생성, 비즈니스 프로세스 표준화를

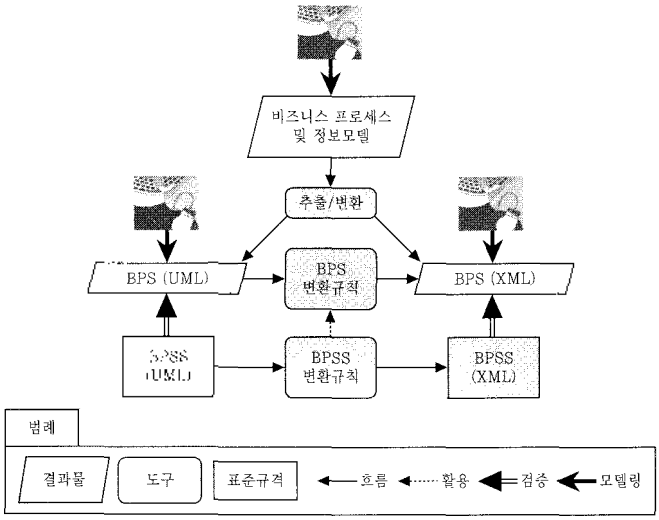
위한 심사 등 다양한 활용이 가능하다.

본 논문에서는 XML 버전 BPSS의 XML Schema에 표현되지 못한 의미 제약들을 체계적으로 발견하고, 이들을 명시적으로 표현하여 재활용하는 방법을 제시한다. 본 논문의 구성은 다음과 같다. 2장에서는 ebXML BPSS에 대한 소개와 함께 XML 기반 BPSS 사례 개발 작업이 가지는 의미를 검토한다. 3장에서는 XML 버전 BPSS 의미 제약의 분류 체계를 제시하고, 이를 기준으로 BPSS에서 추출한 제약을 정리한다. 4장에서는 XML 제약 언어를 이용하여 의미제약을 표현하고 검증하는 방법과 함께 의미 제약의 다양한 활용 가능성에 대하여 검토한다. 마지막으로 5장에서는 결론 및 앞으로의 연구방향을 제시한다.

2. ebXML BPSS와 의미제약

어떤 기업이 거래 기업과의 상호작용을 편리하게 하기 위하여 공유된 역할 (Role), 관계 (Relationship), 의무사항 (Responsibility) 등을 어떻게 수행할 것인지를 상세히 정의한 것이 기업간 거래에서의 비즈니스 프로세스이다 [3]. 전통적인 기업간 전자상거래에서 많이 사용된 EDI는 비즈니스 프로세스를 충분히 고려하거나 개선하지 않고, 단순히 기존의 문서만을 표준화하여 사용하였기 때문에, 문서를 합리화하고 단순화하는 것이 불가능하였다. 최근에는 xCBL [19], UBL [11], OAGIS BOD [10] 등과 같이 XML을 기반으로 하는 문서 표준화 활동도 활발한데, 재활용의 수준을 문서에서 비즈니스 시나리오 (비즈니스 프로세스)로 확대하여, 비즈니스 시나리오를 정의하는 과정에서 문서에 대한 검토가 자연스럽게 이루어질 수 있도록 지원하며, 비즈니스 시나리오 수준에서 기업간 거래를 실행하고 관리될 수 있는 방안이 필요하다. 명확히 정의된 비즈니스 프로세스에 따라서 기업간 거래를 실행하고 관리하는 기능이 지원될 수 있다면, 비즈니스 시나리오를 프로그램 코드에 구현하여 수행하는 기존의 B2B 시스템들 (EDI시스템, XML/EDI시스템 등)이 변화에 유연하게 대응하기 어려웠던 점을 극복할 수 있다.

ebXML 프레임워크에서는 거래 시나리오와 문서를 정의하기 위한 체계적인 분석 작업에 UMM (UN/CEFACT Modeling Methodology)을 사용할 것을 권고하고 있다. UMM은 비즈니스 프로세스와 정보 (문서)의 분석과 정의를 위한 방법론이다. UMM 메타 모델은 UML을 확장하여 ebXML 프레임워크 내에서 비즈니스 프로세스를 분석할 때 도출되어야 하는 모든 정보들의 타입을 제시하고 있다 [17]. BPSS는 UMM 메타 모델의 부분집합으로서, 비즈니스 프로세스를 실제 실행하기 위해서 반드시 필요한 내용만을 규정하는 측면에서 기업간 협업을 정의할 수 있도록 지원하기 위한 것이다. ebXML 프레임워크는 표준 객체지향 모델링 언어인 UML을 기본 모델링 언어로 활용하고 있다. UMM, UMM 메타모델, BPSS, 모두 UML을 활용하거나 이를 기초로 설명되고 있다. 예를 들어, BPSS는 <그림 1>과 같이 UML과 XML Schema, 이 두 가지의 독립적인 표현 방식을 제공하다. BPSS 변환규칙의 집합도 제공되는



〈그림 1〉 ebXML BPS 생성 과정

데, 이것은 UML 버전의 BPSS로부터 XML 버전의 BPSS로의 변환을 정의한다. 또한 이에 근거한 BPS 변환규칙은 UML 버전의 BPS와 XML 버전의 BPS간의 변환을 정의한다.

ebXML에서 BPS의 작성은 크게 3가지 방법이 가능하다. 즉, ebXML 비즈니스 프로세스 분석 워크쉬트와 가이드라인에 기반한 편집기 도구를 사용하는 방법, UML 버전의 BPSS를 사용하는 방법, XML 버전의 BPSS를 사용하는 방법이다. ebXML 비즈니스 프로세스 분석 워크쉬트와 가이드라인은 UMM을 따라 비즈니스 프로세스와 정보 모델을 생성하는 과정을 지원하기 위해서 제공된다[3]. 이에 기반한 편집기 도구를 이용하여 비즈니스 프로세스와 정보 모델을 작성하고, 이중 일부분을 추출하여 BPS를 작성한다. BPS 작성을 위한 다른 방법은 UML 버전의 BPSS 또는 XML 버전의 BPSS를 활용하여 직접 BPS를 생성하는 것이다. 어떠한 과정을 통해서 작성되든지, 궁극적으로 비즈니스 프로세스를 실행하는 측면에서 사용되는 형태인 XML 버전 BPSS 사례가 생성되어야 한다.

XML 버전 BPSS를 준수하여 정의된 BPSS 사례는 비즈니스 프로세스에 필요한 업무 문서의 구조를 정의하거나 기업의 기술적인 능력을 정의하는데 참조되며, 실제로 비즈니스 프로세스를 실행하는 과정에서는 비즈니스 프로세스 엔진을 구동하는 시나리오 역할을 수행하게 된다. 따라서 특정한 비즈니스 프로세스와 관련하여 실존하는 의미 제약을 완전하고 일치되게 정의할 수 있는가 하는 것이 BPS를 (재)활용하는데 있어서 매우 중요하다.

개념적인 수준에서 의미 제약을 표현하여 처리할 수 있다면 여러 가지 장점을 얻을 수 있게 되는데, 일반적으로 작성중인 모델에 대

한 의미 제약을 검증할 수 있고 모델링 작업을 안내할 수도 있다. 모델이 표준 객체지향 모델링 언어인 UML로 작성되는 경우라면, OCL (Object Constraint Language) 기반으로 의미 제약을 표현하는 것도 가능하다 [3]. 이렇게 명시적으로 표현된 의미 제약은 모델링 과정에서 또는 모델링이 모두 완료된 후 사후적으로 모델에 적용될 수 있다. ebDesinger [7]와 같은 편집기 도구에서는 자체 코드의 일부분에 이러한 제약을 구현하여 모델의 정확성이나 완전성을 검증하나, 구현하기가 어렵고, 유연성과 확장성이 부족하다. 규칙 기반의 추론 언어를 이용하여 검증하는 것이 보다 바람직하다 [8].

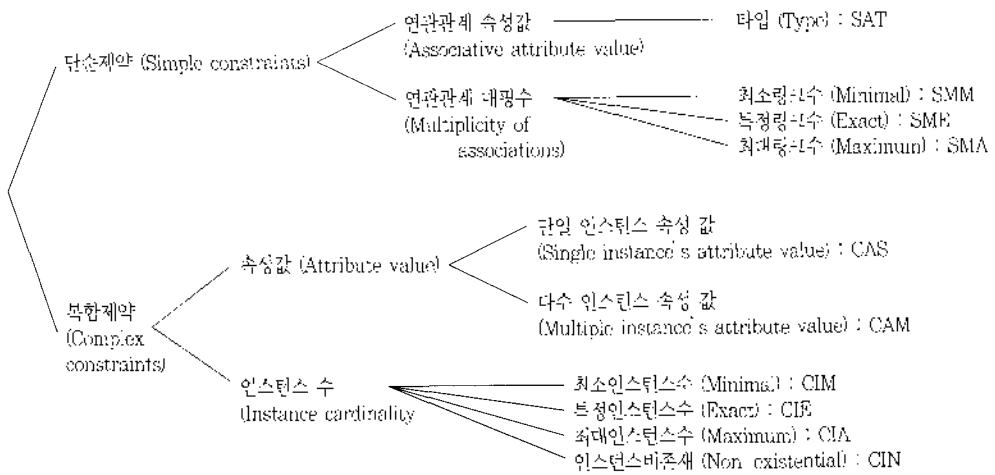
BPS 생성 방법에 상관없이, 최종적으로 실행 가능한 XML 버전의 BPS에 대한 검증이 매우 중요하다. 사용자가 직접 XML로 BPS를 작성한 경우뿐만 아니라, 변환을 통해서 생성된 XML 버전 BPS를 수정한 경우에도 이러한 점검이 반드시 필요하다. XML기반의 BPSS에서는 XML Schema를 이용하여 명세에 관한 제약을 정의하고 있으나, 비즈니스 프로세스에 대한 의미 제약을 모두 표현하기에는 매우 부족하다. 예를 들면, 어떤 요소가 존재하면 또 다른 요소가 존재해야 한다는 것과 같은 의미 제약의 표현은 XML Schema를 이용한다고 하더라도 표현이 불가능하다. 이러한 목적을 달성하기 위해서는 BPS를 읽어들여 분석하는 전용 프로그램을 작성하거나, Schematron [6] 이나 XCSL (XML Constraint Specification Language) [15] 과 같은 XML문서에 대한 의미 제약 표현 언어를 이용하거나, XSLT와 같은 변환 언어를 이용할 수 있다.

3. ebXML BPSS 제약의 구분 및 추출

3.1 제약식의 구분

개념적인 수준에서 BPSS의 의미 제약을 모델링하고 검증하는 방법에 관해서는 김종우와 김형도의 연구[8]가 있다. 여기에서는 Jacinto et al. 의 연구[5]를 바탕으로 제약을 크게 단순제약, 정상적인 표현에 대한 패턴 매칭, 그리고 복합제약으로 구분하고, 이들을 OCL로 표현하고 있다. 단순제약 (Simple constraint) 은 한 속성값 또는 한 연관관계의 매핑수에 국한된 제약 조건을 의미한다. 정상적인 표현에 대한 패턴 매칭 제약 (Pattern matching against a regular expression) 은 속성값의 표현을 특정 표기 기법을 맞추어서 작성해야 하는 제약을 의미한다. 그리고 복합제약 (Complex constraint) 은 두 개 이상의 속성값 또는 두 개 이상의 객체 인스턴스의 상태가 관련된 제약을 의미한다. 이러한 분류는 UML 클래스의 속성 및 연관관계를 기준으로 한 것으로, XML의 엘리먼트 및 속성과는 약간의 차이가 있다. 특히 어떤 엘리먼트가 다른 엘리먼트와 연관되어 있음은 IDREF 유형의 속성을 이용하게 되는데, 이 때 참조되는 엘리먼트의 타입을 XML Schema로 규정하지 못한다. 또한 UML 클래스의 속성은 클래스에 해당되는 엘리먼트의 속성이나 이 엘리먼트에 포함되는 또 다른 엘리먼트로 표현될 수 있다.

이 논문에서는 김종우와 김형도의 연구[8] 결과를 기본으로 하되, XML의 엘리먼트와



〈그림 2〉 ebXML BPSS 제약식의 분류

속성을 중심으로 제약의 종류를 〈그림 2〉와 같이 재정리하였다. 이 그림에서 유의할 점은 XML Schema로 표현이 가능한 제약들은 이러한 분류에서 제외되었다는 것이다. 정상적인 표현에 대한 패턴 매칭 제약의 경우 XML Schema로 표현이 가능하므로 분류에서 제외되었다. 단순제약은 하나의 속성에 대한 제약으로 연관관계 속성값과 연관관계 매핑수로 구분하였다. 여기에서 전자는 연관관계를 표시하는 속성의 값, 즉 또 다른 엘리먼트의 유형에 관한 제약이며, 후자는 이러한 연관관계에서의 매핑수에 관한 제약이다. 〈그림2〉에서 나무의 가지에 제시된 SAT 등의 약어는 제약식의 종류를 구분하기 위한 것이다. SAT의 경우는 단순제약 (Simple constraints), 연관관계 속성값 (Associative attribute value), 그리고 타입 (Type)을 결합한 약어이다.

3.2 제약식의 추출

BPSS는 크게 4가지 (Business Collaborations, Business Transactions, Document Flow, Choreography) 부분으로 구분하여 제시되고 있으며, 각각에 속한 구체적인 구성요소는 다음 〈표 1〉과 같다.

BPSS에 관한 제약으로는 ebXML BPSS 공식문서 (현재 버전 1.10) 에서 일반적인 BPSS 의미 제약으로 제시된 13개의 WFR (Well-Formedness Rule), XML 버전 BPSS에서 엘리먼트 별로 언급된 7개의 WFR과 함께, 김종우와 김형도의 연구[8]에서 추가적으로 확인된 14개의 WFR이 있다. 이러한 WFR들은 XML Schema로 표현되지 않는 제약만을 대상으로 한다. 이러한 제약들을 보다 상세하게 분석하여 〈표 1〉에 정리된 XML 엘리먼트 별로 다시 정리한 것이 〈표 2〉이다. 이 표에서

〈표 1〉 XML버전 ebXML BPSS의 주요 엘리먼트

구 분	의 미	주요 엘리먼트
양자간 협업	두 거래 참여자간에 이루어지는 비즈니스 협업으로 각각의 참여자는 하나의 Role을 수행한다. 비즈니스 협업의 상태는 BusinessActivity들로 표현된다. BusinessActivity는 BusinessTransaction을 수행하는 행위 (BusinessTransactionActivity) 나 또 다른 BinaryCollaboration을 수행하는 복합적인 행위 (CollaborationActivity) 를 말한다.	BinaryCollaboration, BusinessActivity, Role, BusinessTransactionActivity, CollaborationActivity
다자간 협업	3개 이상의 BusinessPartnerRole들간에 이루어지는 BinaryCollaboration들을 통합하여 비즈니스 협업을 구성한다. 각각의 BusinessPartnerRole은 어떤 BinaryCollaboration에서 권한이 부여된 Role들 중 하나를 수행하거나, 또는 다수의 양자간 협업에서 각각 하나의 권한이 부여된 역할을 수행함을 의미한다. 이러한 관계는 Performs로 표현된다.	MultiPartyCollaboration, BusinessPartnerRole, Performs
비즈니스 거래와 문서 흐름	BusinessTransaction은 두 거래 참여자간의 단위 작업이며, 두 가지의 비즈니스 행위와 하나 이상의 문서 흐름으로 구성된다. RequestingBusinessActivity에는 반드시 하나의 문서 흐름을 동반하게 되며, RespondingBusinessActivity에는 다수의 문서 흐름을 동반할 수도 있으며, 전혀 없을 수도 있다. 문서 흐름은 직접적으로 정의되는 대신에, 두 거래 참여자의 역할간에 전송되는 DocumentEnvelope를 사용하여 간접적으로 정의된다. 하나의 DocumentEnvelope에는 하나의 BusinessDocument만이 포함되며, 이 문서와 관련된 다수의 Attachment가 동반되어 포함될 수 있다.	BusinessTransaction, BusinessAction, RequestingBusinessActivity, RespondingBusinessActivity, DocumentSecurity, DocumentEnvelope, BusinessDocument, Attachment, ConditionExpression
협업안무	특정한 BinaryCollaboration에 통합된 BusinessTransaction이나 또 다른 BinaryCollaboration들을 표현하는 BusinessActivity들 / 특정한 MultiPartyCollaboration에 통합된 BinaryCollaboration들을 표현하는 BusinessActivity들을 정렬시키고, 순서화시키는 것으로, BusinessState와 이들간의 Transition을 사용하여 정의된다. BusinessActivity가 BusinessState의 일종으로 표현되어 전체적인 협업안무가 작성되며, 보조적으로 Start, Success, Failure, Fork, Join, Decision 등 다수의 BusinessState가 제공된다.	BusinessState, Transition, Start, CompletionState, Success, Failure, Fork, Join, Decision

〈표 2〉 XML 버전 BPSS의 엘리먼트별 의미제약

엘리먼트	WFR 번호	WFR의 내용	제약의 종류
Requesting Business Activity	0	부인방지 (Non-repudiation) 가 요구되는 경우에는 문서흐름의 변조검출 (Tamper-detection) 이 가능해야 한다.	CAM
	1	권한확인 (Authorization) 이 요구되는 경우에 문서흐름과 업무신호는 신원확인 (Authentication) 이 되거나, 변조검출 (Tamper-detection) 이 가능해야 한다.	CAM
	2	수신확인시간 (timeToAcknowledgeReceipt) 과 수락확인시간 (timeToAcknowledgeAcceptance) 이 모두 값을 가지고 있는 경우에는 수신확인시간이 수락확인시간보다 적어야 한다.	CAS
	3	수락확인시간이 없으면, 행위수행시간은 수신확인시간과 같거나 더 커야 한다.	CAS
	5	수신 부인방지가 요구되는 경우에 수신확인시간은 반드시 설정되어야 한다.	CAS
	6	수신확인시간, 수락확인시간, 행위수행시간이 모두 0이 될 수는 없다.	CAM
	28	긍정적응답여부 (isPositiveResponse) 는 요청업무행위 (RequestingBusinessActivity) 에 의해서 송신되는 문서봉투 (DocumentEnvelope) 에는 적절하지 않다.	CAS
RespondingBusinessActivity	0	부인방지가 요구되는 경우에는 문서흐름의 변조검출이 가능해야 한다.	CAM
	1	권한확인 (Authorization) 이 요구되는 경우에 문서흐름과 업무신호는 신원확인 (Authentication) 이 되거나, 변조검출 (Tamper-detection) 이 가능해야 한다.	CAM
	2	수신확인시간 (timeToAcknowledgeReceipt) 과 수락확인시간 (timeToAcknowledgeAcceptance) 이 모두 값을 가지고 있는 경우에는 수신확인시간이 수락확인시간보다 적어야 한다.	CAS
	5	수신 부인방지가 요구되는 경우에 수신확인시간은 반드시 설정되어야 한다.	CAS
	29	수신확인시간 또는 수락확인시간이 설정되어 있으면, 적어도 하나의 응답 문서봉투가 정의되어 있어야 한다.	CIM
Business Activity	7	완료상태는 상호 배타적으로 정의되어, 단 하나의 완료상태만이 도달되도록 보장하여야 한다.	CIE
	8	업무행위 (BusinessActivity) 다수의 입력 전이 (Transition) 를 가질 수 있으나, 출력 전이는 단 하나여야 한다. 하나 이상의 출력 전이를 규정하기 위해서는 분지 (Fork) 또는 의사결정 상태를 이용해야 한다.	CIM CIE
MultiParty Collaboration	13	모든 다자간 협업 (MultiPartyCollaboration) 은 양자간 협업으로부터 조립되어야 한다.	CAM
Business PartnerRole	14	주어진 업무행위에서 한 참여자는 두 가지 역할을 모두 수행할 수 없다.	CAM

Performs	15	어떤 역할을 수행하는 모든 Performs에는 반대되는 역할을 수행하는 Performs가 있어야 한다.	CIM
Document Envelope	16	각각의 문서봉투는 하나의 요청행위와 하나의 응답행위와 관련되어 있다.	CIE
	17	긍정적응답여부(isPositiveResponse) 는 요청업무행위(RequestingBusinessActivity) 에 의해서 송신되는 문서봉투(DocumentEnvelope) 에는 적절하지 않다.	CAM
Success	18	모든 양자간 협업 (BinaryCollaboration) 에는 하나 이상의 성공 (Success) 상태가 있어야 한다.	CIM
Failure	19	모든 양자간 협업에는 하나 이상의 실패 (Failure) 상태가 있어야 한다.	CIM
Binary Collaboration	9	양자간 협업에는 최대 하나의 시작 (Start) 상태가 있어야 한다.	CIA
	10	양자간 협업에는 적어도 하나의 완료 상태가 있어야 한다.	CIM
	12	양자간 협업과 관련된 2개의 역할은 상이해야 한다.	CAS
	20	양자간 협업은 협업 행위를 통해서 자기 자신을 재사용해서는 안 된다.	CIN
	21	양자간 협업의 initiatingRoleIDREF 속성은 이 협업의 두 가지 역할 중 하나를 참조해야 한다.	CAM
Business Transaction Activity	4	응답이 없는 요청으로 설정되지 않는 한, 거래수행시간은 반드시 설정되어야 한다.	CAM
	11	주어진 업무거래행위의 두 가지 역할이 같을 수는 없다.	CAS
	22	하나의 업무거래행위는 businessTransaction 속성으로 비즈니스 거래를 참조하여야 한다.	SAT
	23	주어진 업무거래행위의 fromRole과 toRole 속성은 업무거래행위에 의해서 수행되는 양자간 협업의 두 역할 중 하나와 일치되어야 한다.	SAT
Collaboration Activity	24	하나의 협업행위는 binaryCollaboration 속성으로 양자간 협업을 참조하여야 한다.	SAT
	25	협업행위에 의해서 수행되는 양자간 협업의 두 역할 중 하나와 일치되어야 한다.	SAT
	26	주어진 협업행위의 fromRole과 toRole 속성은 같은 값을 가질 수 없다.	CAS
Document Envelope	27	하나의 문서봉투는 정확히 하나의 주된 업무 문서를 포함한다.	CIE
Start	30	시작 (Start) 의 toBusinessState 속성은 양자간 협업의 한 행위를 참조하여야 한다.	SAT
Success	31	성공 (Success) 의 fromBusinessState 속성은 양자간 협업의 한 행위를 참조하여야 한다.	SAT
Failure	32	실패 (Failure) 의 fromBusinessState 속성은 양자간 협업의 한 행위를 참조하여야 한다.	SAT
Transition	33	전이 (Transition) 의 fromBusinessState와toBusinessState 속성은 양자간 협업의 한 행위를 각각 참조하여야 한다	SAT

WFR 번호는 김종우와 김형도의 연구[8]에서 의미제약의 확인자로 사용된 것인데, XML 엘리먼트 별로 WFR이 재정리되면서 여러 엘리먼트에 중복되어 나타나는 경우도 있다.

4. 제약의 표현과 검증

4.1 BPSS 의미제약의 표현 및 검증 방법

오늘날 의미를 검증하는 가장 일반적인 방법은 응용의 일부분으로 검증 기능 프로그램을 구현하는 것이다. 이것은 선언적인 문법 검증이 불가능한 경우로, 여러 가지 단점을 동반한다. 프로그래머들은 검증 코드를 작성해야 하므로, 검증 대상 문서의 유형에 친숙해져야 한다. 검증 코드는 XML 데이터를 접근하는 코드들과 뒤섞이게 되고, 결국 검증에 집중할 수 없도록 만든다. 프로그램 언어에 종속적으로 검증 코드가 작성되어 이식성이 낮고, 재활용 및 관리도 어렵다. 이러한 모든 요인들이 의미 검증을 어렵게 하고, 중요한 비즈니스 지식을 비가시적으로 만들게 된다. 프로그램에 의한 검증의 유일한 장점은 처리 성능의 향상이라고 할 수 있는데, 여러 단점들이 이러한 장점을 압도한다. 따라서 이 방법은 본 논문의 고려대상에서 제외된다.

XSLT도 XML 문서의 검증을 위해서도 사용되고 있다. 어떤 엘리먼트에 적용되는 템플릿 규칙을 작성할 수 있으며, XSLT 함수를 이용하여 템플릿에 검증 코드를 삽입할 수 있다. XSLT의 출력은 입력 문서에 대한 진단 메시지를 포함하는 보고서 형태이다. XSLT는 프로그램에 의한 방법보다 장점이 있는데, 선언적인 언어여서 XML 처리 코드를 작성해야 하는 부담을 제거할 수 있다는 것이 가장 중요하다. 그러나 이것은 제약언어가 아니라 변환 언어이기 때문에 몇 가지 단점을 가지고 있다. XSLT의 풍부함이 거꾸로 최적화를 제약한다. 이에 반해서 전용 제약 언어는 표현 재활용이 용이하며, 제약 위반을 해결하는 방안의 수립 등과 같은 제약의 분석도 가능하다. XSLT는 이러한 분석이 불가능하거나 구현하기 어려우며, 문서간 또는 문서와 데이터베이스간 검증도 어렵다.

Schematron[6]은 XPath 기반의 제약언어로서, 불리언 논리를 이용하여 XML 문서의 제약을 표현할 수 있다. Schematron의 제약은 전용 검증 엔진에 의해서 평가되거나, XSLT 템플릿으로 변환되어 XSLT 처리기에 의해서 실행될 수 있다. Schematron이 일견 XSLT와 비슷하지만, 변환을 지원하기 위해 필요한 요소가 생략되어 XSLT보다 범위가 훨씬 좁은 언어이다. 따라서 언어를 사용하기 쉽고, 분석 도구를 작성하기 쉽다. 향후 ebXML 표준화가 진행됨에 따라 새로운 BPSS가 배포되면, 이에 따른 의미 제약의 변화에 신속하게 대응하는 것도 가능하다. XCML[4], XCSL[6], xlinkit[9] 등과 같은 언어도 표현력이나 제공하는 기능이 Schematron과 거의 유사하나, 광범위하게 사용되지는 못하고 있다. 따라서, 이 논문에서는 기본적으로 Schematron을 사용하여 XML기반 BPSS의 의미 제약을 표현하고, 이를 기반으로 의미제약을 검증하고자 한다.

4.2 BPSS 의미제약의 표현

〈그림 3〉은 사례로서 WFR 2, WFR 18, WFR 20을 Schematron으로 표현한 것이다. Schematron의 표현력을 설명하기 위해서 선택된 이들 세 가지 WFR은 모두 복합제약에 해당되며, 속성값에 관한 제약의 일종인 CAS에 속하는 것 (WFR 2와 WFR20) 과, 인스턴스 수에 관한 제약의 일종인 CIM에 속하는 것 (WFR 18) 이 있다.

Schematron에서 규칙(rule)들은 패턴(pattern)으로 그룹화될 수 있는데, 여기서는 개개의 WFR을 하나의 패턴으로 설정하고 있다. 규칙은 여러 개의 assert와 report를 자녀로서 가지고 있으며, 이들에 대한 문맥을 설정하는 context 속성을 제공하여 assert와 report

```
<schema xmlns="http://www.ascc.net/xml/schematron">
  <pattern name="WFR 2">
      <rule context="//BusinessTransaction">
          <assert test="RequestingBusinessActivity/@timeToAcknowledgeReceipt and
                        RequestingBusinessActivity/@timeToAcknowledgeAcceptance and
                        RequestingBusinessActivity/@timeToAcknowledgeReceipt <
                        RequestingBusinessActivity/@timeToAcknowledgeAcceptance">
              This BPS violates WFR 2 (The time to acknowledge receipt must be less than
              the time to acknowledge acceptance if both properties have values.)
          </assert>
          <assert test="RespondingBusinessActivity/@timeToAcknowledgeReceipt and
                        RespondingBusinessActivity/@timeToAcknowledgeAcceptance and
                        RespondingBusinessActivity/@timeToAcknowledgeReceipt <
                        RespondingBusinessActivity/@timeToAcknowledgeAcceptance">
              This BPS violates WFR 2 (The time to acknowledge receipt must be less than
              the time to acknowledge acceptance if both properties have values.)
          </assert>
      </rule>
  </pattern>
  <pattern name="WFR 18">
      <rule context="//BinaryCollaboration">
          <assert test="count(Success)>1">
              This BPS violates WFR 18 (Every BinaryCollaboration should have at least one Success.)
          </assert>
      </rule>
  </pattern>
  <pattern name="WFR 20">
      <rule context="//BinaryCollaboration">
          <assert test="count(CollaborationActivity[@binaryCollaboration=../@name])=0">
              This BPS violates WFR 20 (A BinaryCollaboration may not re use itself
              through a CollaborationActivity.)
          </assert>
      </rule>
  </pattern>
  ...
</schema>
```

〈그림 3〉 Schematron을 이용한 BPSS 의미제약 표현

의 표현을 간결하게 작성할 수 있도록 지원한다. assert는 반드시 충족되어야 할 사항에 대해서 기술하기 위한 것으로, test 속성의 값으로 표현된 XPath 표현이 만족되지 못하면, 그 엘리먼트의 내용을 출력으로 내보낸다. 이와 반대로, report는 오류 사항에 대해서 기술하기 위한 것으로, test 속성의 값으로 표현된 XPath 표현이 만족되면, 그 엘리먼트의 내용을 출력으로 내보낸다.

WFR 2의 경우를 보면 하나의 규칙으로 표현되어 있는데, 이 규칙은 임의의 BusinessTransaction 엘리먼트(//BusinessTransaction)에 적용되는 것으로서, 두 개의 assert를 포함하고 있다. 첫 번째 assert의 경우 RequestingBusinessActivity에서 @timeToAcknowledgeReceipt가 @timeToAcknowledgeAcceptance보다 작아야 함을 규정하고 있으며, 두 번째 assert에서는 RespondingBusinessActivity에서 동일한 내용을 규정하고 있다. 이러한 두 가지를 만족시키지 못할 경우 WFR 2를 위반한 것으로 출력을 내보내게 된다. WFR 18의 경우에는 임의의 BinaryCollaboration 엘리먼트 아래에는 반드시 하나 이상의 Success 엘리먼트가 있어야 함을 규정하고 있다.

WFR 20의 경우에는 Schematron 표현력의 한계를 잘 보여주고 있다. WFR 20의 규칙은 실제로 양자간 협업이 직접 자기 자신을 재사용하는 경우와 또 다른 양자간 협업을 통해서 간접적으로 재사용하는 경우로 나누어 생각해 볼 수 있다. Schematron을 이용해서 간접 재사용 여부를 유추하는 일반적인 표현은 불가능하다. 이에 따라서, <그림 3>에서는 전자의 경우만을 표현하고 있으며, WFR 20에 대한 완벽한 표현이 되지 못한다.

총 34개의 WFR를 모두 검토한 결과, WFR 20을 포함해서 WFR 4, WFR 8, WFR 22, 이 네 개의 WFR은 Schematron을 이용한 정확한 표현이 불가능하였다. 그 이유는 복잡한 변수 설정과 반복적인 처리 등이 불가능하다는 점에 기인한다. 그래서, 이 네 가지 WFR은 <그림 4>와 같이 XSLT를 이용하여 표현되었다. XSLT에서 규칙은 template 엘리먼트로 표현되는데 여기서 규칙의 조건 부분을 표현하는 것이 match 속성이다. WFR 8의 경우를 보면, 임의의 BusinessTransactionActivity나 CollaborationActivity에 대해서 규칙이 적용됨을 알 수 있다. 그리고 다수의 변수(variable)가 사용되는데, 처음 두 개의 변수(nameBA와 nameIDBA)는 조사대상으로 선정된 BusinessTransactionActivity나 CollaborationActivity의 name과 nameID의 값으로 설정된다. 그리고 나머지 3개의 변수(numTransitionFrom, numSuccess, numFailure)는 nameBA 또는 nameIDBA가 일치하는 Transition, Success, Failure 엘리먼트의 갯수를 각각 나타낸다. 최종적으로 이들 3개의 변수 값을 더하여 0이라면 WFR 8 위반임을 알리게 되는 것이다.

<그림 4>에서 WFR 20은 두 개의 template (WFR_20과 WFR_20_1)으로 표현된다. 첫 번째 template은 각각의 BinaryCollaboration에 대하여 한번씩 두 번째 template를 호출한다. 이 때 src라고 하는 패러미터를 BinaryCollaobration의 @name으로 초기화하여 두 번째 template에 전달한다. 두 번째

```
<xsl:template name="WFR_1" match="//BusinessTransaction">
    <xsl:variable name="nameBT" select="@name" />
    <xsl:variable name="nameBTIDREF" select="@nameID_REF" />
    <xsl:if test="(RequestingBusinessActivity/@timeToAcknowledgeReceipt and
                  RequestingBusinessActivity/@timeToAcknowledgeReceipt &gt; 0) or
                  (RequestingBusinessActivity/@timeToAcknowledgeAcceptance and
                  RequestingBusinessActivity/@timeToAcknowledgeAcceptance &gt; 0)">
        <xsl:for-each select="//BusinessTransactionActivity[@businessTransaction=$nameBT or
            @businessTransactionIDREF=$nameBTIDREF]">
            <xsl:if test="not(@timeToPerform)">
                This BPS violates WFR 4 (The time to perform a transaction cannot be null
                unless it is specified to be request without a response.)
            </xsl:if>
        </xsl:for-each>
    </xsl:if>
</xsl:template>

<xsl:template name="WFR_8" match="//BusinessTransactionActivity | //CollaborationActivity">
    <xsl:variable name="nameBA" select="./@name" />
    <xsl:variable name="nameIDBA" select="./@nameID" />
    <xsl:variable name="numTransitionFrom"
        select="count(../Transition[@fromBusinessState=$nameBA or @fromBusinessStateIDREF=$nameIDBA])" />
    <xsl:variable name="numSuccess"
        select="count(../Success_@fromBusinessState=$nameBA or @fromBusinessStateIDREF=$nameIDBA')" />
    <xsl:variable name="numFailure"
        select="count(../Failure_@fromBusinessState=$nameBA or @fromBusinessStateIDREF=$nameIDBA')" />
    <xsl:if test="$numTransitionFrom+$numSuccess+$numFailure=1">
        This BPS violates WFR 8 (A BusinessActivity may have
        any number of incoming transition but only one output transition.)
    </xsl:if>
</xsl:template>

<xsl:template name="WFR_20" match="//BinaryCollaboration">
    <xsl:call-template name="WFR_20_1"><xsl:with-param name="src" select="@name"/></xsl:call-template>
</xsl:template>

<xsl:template name="WFR_20_1">
    <xsl:param name="src" select="..." />
    <xsl:for-each select="CollaborationActivity">
        <xsl:variable name="binaryCollaboration" select="@binaryCollaboration" />
        <xsl:choose>
            <xsl:when test="$src=@binaryCollaboration">This BPS violates WFR 22
                (A BinaryCollaboration may not re-use itself through a CollaborationActivity.)
            </xsl:when>
            <xsl:otherwise>
                <xsl:for-each select="//BinaryCollaboration[@name=$binaryCollaboration]">
                    <xsl:call-template name="WFR_20_1">
                        <xsl:with-param name="src" select="$src" />
                    </xsl:call-template>
                </xsl:for-each>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:for-each>
</xsl:template>

<xsl:template name="WFR_22" match="//BusinessTransactionActivity">
    <xsl:variable name="refBT" select="./@businessTransaction" />
    <xsl:variable name="refBTIDREF" select="./@businessTransactionIDREF" />
    <xsl:variable name="numBT"
        select="count(//BusinessTransaction[@name=$refBT and local-name(.)='BusinessTransaction'])" />
    <xsl:if test="$numBT!=1">
        <xsl:if test="not($refBTIDREF) or id($refBTIDREF)[position()=1 and local-name(.)!='BusinessTransaction']">
            This BPS violates WFR 22 (A BusinessTransactionActivity must refer to
            a BusinessTransaction by the businessTransaction attribute.)
        </xsl:if>
    </xsl:if>
</xsl:template>
```

〈그림 4〉 XSLT를 이용한 제약의 표현

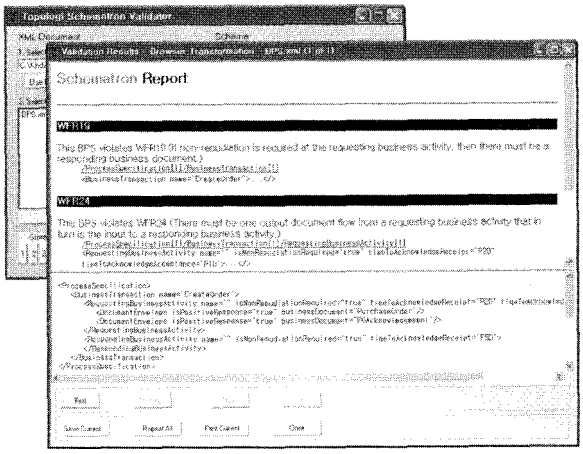
template은 CollaborationActivity에서 참조된 @binaryCollaboration하고 src 패러미터의 값과 동일한 것인지 여부를 검사한다. 동일하다면 오류 메시지를 출력하고, 그렇지 않으면, 추가적인 검사를 실행한다. BinaryCollaboration은 여러 BinaryCollaboration들이 다계층 구조로 구성될 수 있으므로, CollaborationActivity의 @binaryCollaboration이 지칭하는 BinaryCollaboration을 찾아서 이 template를 다시 호출하게 된다.

4.3 BPSS 의미 제약의 검증

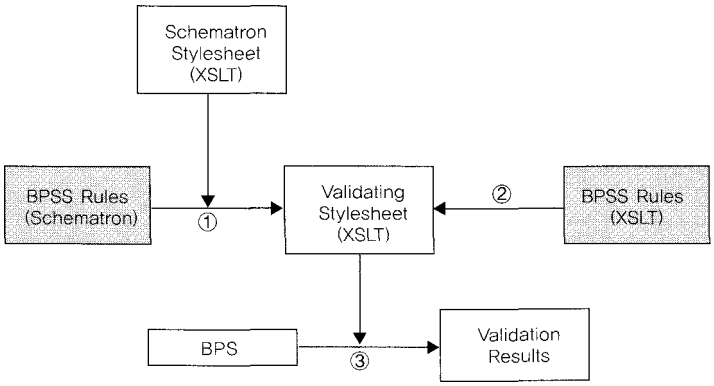
Schematron으로 작성된 대부분의 BPSS 의미 제약은 Schematron을 지원하는 소프트웨어를 이용하여 검증이 가능하다. <그림 5>는 Schematron을 지원하는 공개 소프트웨어 [16]를 이용해서 검증하는 경우를 보여준다.

그러나, Schematron으로 표현되지 못한 WFR을 함께 검증하기 위해서, <그림6>과 같이 XSLT를 추가적으로 이용하는 방법을 선택하였다. Schematron을 이용해서 표현된 BPSS 의미제약은 Schematron 스타일쉬트(XSLT)를 이용하여 XSLT 템플릿으로 변환되었다 (<그림6>의 ①). 그리고 여기에 처음부터 XSLT를 이용해서 표현한 제약들을 더해서 검증을 위한 스타일쉬트(XSLT)를 완성하였다 (<그림6>의 ②). 그리고 이것을 이용하여 BPS 검증 작업이 이루어질 수 있도록 구현하였다 (<그림6>의 ③).

<그림 7>은 웹 브라우저를 이용하여 XSLT로 BPS가 BPSS의 의미제약을 만족하는지 알아볼 수 있는 응용을 보여준다. 먼저 검증하고자 하는 BPS 파일을 선택하고, 이 파일에 적용하고자 하는 BPSS 제약사항을 표현한 XSLT 파일을 선택한다. 여기서는 BPS 파일로는 Sample_BPS.xml이 선택되었고, BPSS 의미제약을 가지고 있는 파일로는



<그림 5> Schematron을 이용한 XML 의미제약 검증

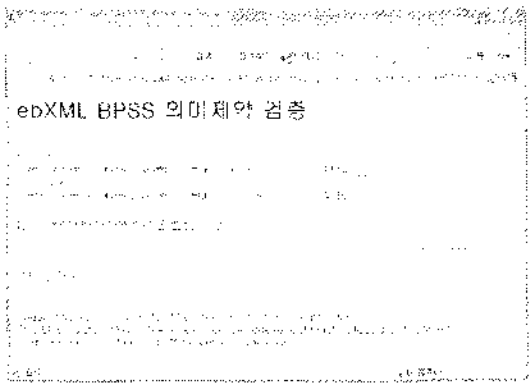


〈그림 6〉 BPS 검증과정

BPSS_Constraints.xsl 이 설정되었다. BPS 검증 버튼을 누르면 이에 따른 검증 위반 사항이 아래 부분에 제시된다.

4.4 토의

XML 제약표현언어를 이용하면 XML 문서에 대한 정확한 제약의 표현이 가능하다. 이를 이용해서 BPSS 의미제약을 표현하고 검증하는 작업은 매우 간편하다. 그러나, XML로 표현된 복잡한 의미제약의 경우 초보자가 이해하기 어렵고 표현하기도 쉽지 않은 단점이 있기 때문에, 개념적 수준에서의 의미제약의 표현과 활용이 바람직스럽다. 그러나, 비즈니스 프로세스 실행을 위해서는 최종적으로 XML기반의 BPS가 생성되어야 하



〈그림 7〉 XSLT 기반의 BPSS 사례 의미제약 검증

고, 비즈니스 프로세스 및 정보 모델이나 UML기반의 BPS을 변환해서 자동 생성한 경우라 하더라도 수정이 불가피한 경우가 있으며, XML에 익숙한 사용자는 직접 XML기반의 BPS를 작성할 수도 있기 때문에, XML 수준에서의 제약의 표현과 검증도 반드시 필요하다.

XML 제약표현언어를 이용하는 방법은 UML 버전의 BPS나 비즈니스 프로세스 및 정보 모델 수준에서 의미제약을 표현하는데 있어서도 유용하다. 왜냐하면 UML로 표현된 내용은 XMI[13]라고 하는 XML 언어로 저장되는 경우가 많기 때문에, 개념적인 수준에서의 의미제약을 XMI에 대한 제약으로 표현할 수 있기 때문이다.

Schematron, XCSL, xlinkit 등 XML 제약표현언어로 명세된 제약들을 각 제약언어별로 제시된 검증 방안이나 도구를 사용하여 오류 점검이 가능하다. 그러나, 제약언어의 표현력에 한계가 있고, 표준화가 미비하기 때문에, XSLT와 같은 일반적인 XML 변환 언어에 의해서 처리하는 것이 때로는 불가피하다. 규칙 엔진을 이용하여 검증하면 반복적인 추론에 의한 의미제약의 검증이 쉽게 가능한데, 이를 위해서는 CLIPS[1]와 같은 규칙 표현언어를 이용할 수 있도록 의미제약의 표현을 변환하는 작업이 선행되어야 한다 [8].

본 연구와 같이 제약들의 정형화 노력을 통해서 BPSS 표준화에 명확한 기준을 제공할 수 있고, 궁극적으로 BPS의 의미적 완성도를 높일 수 있다. 이렇게 발견되고 정의된 BPSS 의미제약들은 BPS의 검증뿐만 아니라, 모델링 과정에서 제약에 어긋나는 부분에 대한 적극적인 수정을 가능하게 할 것이며, BPSS 표준에 대한 일치성 (Conformance) 을 확인하기 위한 자료의 생성 등에도 유용하게 활용될 수 있다.

5. 결 론

본 연구에서는 먼저 ebXML BPSS 표준에 포함된 의미 제약들을 XML 구성요소 별로 정리하였다. 이러한 작업은 BPSS 표준에서 일부 정의된 내용을 참조하고, XML 버전의 BPSS를 사용하여 검증이 불가능한 부분을 추가하였다. 이러한 작업의 결과는 구체적으로 국내외의 업종별 비즈니스 프로세스 표준화 과정에서 생성된 XML 버전 BPSS 사례들을 검증하는 작업에 직접적으로 활용할 수 있으며, 적극적인 방법으로 모델링 과정을 안내하거나 비즈니스 프로세스 관리 시스템이 표준을 제대로 지원하는지 알기 위한 일치성 (Conformance) 테스트 입력 자료를 생성하는 등의 다양한 용도로 활용 가능하다. 정리된 의미 제약은 XML 기반의 제약언어와 XSLT를 사용하여 명시적으로 표현되었다. 이렇게 정의된 의미 제약을 이용하여 검증하는 과정을 데모하기 위해서 웹 브라우저 기반으로 검증을 위한 XSLT 파일을 BPS에 적용할 수 있는 응용을 제시하였다.

이러한 방법은 개념적인 수준에서 이루어지는 것과 보조적인 것으로 실무적인 관점에서는 매우 중요한 일이다. 추후 의미제약을 모두 표현할 수 있는 XML 제약 표현언어를 개발하는 것과 개념적인 수준의 제약을 XML

수준의 제약으로 변환하여 적용하는 방법을 연구할 계획이며, 지능형 BPS 설계도구에서 적극적으로 의미제약을 활용할 수 있는 방안을 탐구할 계획이다.

참 고 문 헌

[1] CLIPS. CLIPS Reference Manual (Version 6.20). http://www.ghg.net/clips/CLIPS.html. 2002.

[2] ebXML. Business Process Analysis and Guidelines v1.0. May 2001.

[3] ebXML. "Enabling a Global Electronic Market." http://www.ebxml.org. 2005.

[4] Jinkun Hu and Lixin Tao. "An Extensible Constraint Markup Language: Specification, Modeling, and Processing." Proceedings of the XML 2004 Conference. Washington D.C., November 2004.

[5] M.H. Jacinto, G.R. Librelotto, J.C. Ramalho, P.R. Henrigues. "Constraint Specification Languages: Comparing XCSL, Schematron, and XML-Schemas." Proceedings of the XML Europe 2002. Barcelona, May 2002.

[6] R. Jelliffe. "The Schematron: An XML Structure Validation Language using Patterns in Trees." http://www.ascc.net/xml/resource/schematron/schematron.html. 2002.

[7] HyoungDo Kim. "Conceptual Modeling and Specification Generation for B2B Business Process Based on ebXML." ACM SIGMOD Record. Vol. 31, No. 1, March 2002.

[8] Jong Woo Kim and Hyoung Do Kim. "Semantic Constraint Specification and Verification of ebXML Business Process Specifications." Expert Systems with Applications, Vol. 27, 2004, pp. 571-584.

[9] C. Nentwich, L. Capra, W. Emmerich and A. Finkelstein. "xlinkit: A Consistency Checking and Smart Link Generation Service." ACM Transactions on Internet Technology. Vol. 2, No. 2, pp. 151-185, May 2002.

[10] OAGi. Open Applications Group. http://www.openapplications.org/, 2002.

[11] OASIS. OASIS Universal Business Language TC. http://www.oasis-open.org/committees/ubl/, 2002.

[12] OMG. Unified Modeling Language Specification (Version 1.3). http://www.omg.org/, 1999.

[13] OMG. XMI. Metadata Interchnage (XMI). http://www.omg.org/, 1999.

[14] J. Rumbaugh, I. Jacobson, G. Booch. The Unified Modeling Language Reference Manual. Wesley, 1999.

[15] J.C. Ramalho. "Constraining Content: Specification and Processing." XML Europe 2001, 2001.

[16] Topologi Pty. Ltd.. "Schematron

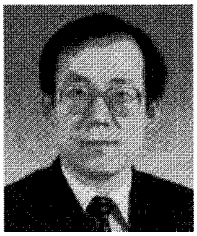
Validator," http://www.topologi.com/products/validator/index.html. 2003.

[17] UN/CEFACT, UN/CEFACT's Modeling Methodology (UMM).http://www.untmg.org/doc_bpwg.html. November 2001.

[18] UN/CEFACT, ebXML Business Process Specification Schema (Version 1.10), Oct. 2003.

[19] xCBL.org, XML Common Business Library, http://www.xcbl.org/, 2002

저 자 소 개



김형도 (E-mail: hdkim@hycu.ac.kr)

1985. 2. 서울대학교 산업공학과 졸업 (학사)

1987. 2. 한국과학기술원 경영과학과 졸업 (석사)

1992. 8. 한국과학기술원 경영과학과 졸업 (박사)

1993 ~ 1999. ㈜데이콤 EC인터넷 연구/기술 팀장

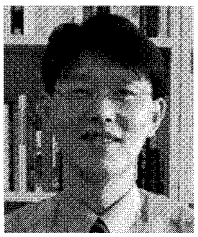
2000 ~ 2002. 아주대학교 정보통신전문대학원 교수

2002 ~ 현재 전자상거래표준화통합포럼 전자문서기술위원회 부위원장

2003 ~ 현재 한양사이버대학교 경영정보학과 교수

2004 ~ 현재 ebXML 전문위원회 위원장

관심분야 XML, 전자상거래, 비즈니스 프로세스, 디지털 워터마킹, 아키텍쳐, 객체지향모델링, 데이터 마이닝 등



김종우 (E-mail: kjw@hanyang.ac.kr)

1989. 2. 서울대학교 수학과 (이학사)

1991. 2. 한국과학기술원 경영과학과 (공학석사)

1995. 8. 한국과학기술원 산업경영학과 (공학박사)

1995. 8. ~ 1996. 8. 한국과학기술원 첨단경영정보연구센터 연수연구원

1996. 9. ~ 2003. 2. 충남대학교 통계학과 전임강사, 조교수, 부교수

1999.12. ~ 2000.12. University of Illinois at Urbana-Champaign Visiting Scholar

2003. 3. ~ 현재 한양대학교 경영학부 부교수

관심 분야 전자상거래, 비즈니스 프로세스 모델링, 상품추천기술, 데이터마이닝 응용