韓國IT서비스學會誌
第12卷 第3號
2013年 9月, pp.311-321
http://dx.doi.org/10.9716/KITS.2013.12.3.311

Linked Data 기반의 메타데이타 모델을 활용한 소프트웨어 모델 통합

김대환* · 정찬기**

Software Model Integration Using Metadata Model Based on Linked Data

Dae-Hwan Kim* · Chan-Ki Jeong**

■ Abstract ■

In the community of software engineering, diverse modeling languages are used for representing all relevant information in the form of models. Also many different models such as business model, business process model, product models, interface models etc. are generated through software life cycles. In this situation, models need to be integrated for enterprise integration and enhancement of software productivity. Researchers propose rebuilding models by a specific modeling language, using a intemediate modeling language and using common reference for model integration. However, in the current approach it requires a lot of cost and time to integrate models. Also it is difficult to identify common objects from several models and to update objects in the repository of common model objects. This paper proposes software model integration using metadata model based on Linked data. We verify the effectiveness of the proposed approach through a case study.

Keyword : Modeling language, Model Integration, Linked Data, Metadata

논문투고일 : 2013년 07월 26일 논문수정완료일 : 2013년 09월 08일 논문게재확정일 : 2013년 09월 13일
* 국방대학교 국방관리대학원 국방과학학부, 교신저자
** 국방대학교 국방관리대학원 국방과학학부

1. 서 론

OMG(Object Management Group)는 객체지향 분석 및 설계의 표준 모델링 언어로 UML(Unified Modeling Language)을 지정하였으며 산업계의 소프트웨어 개발 표준 도구로 사용되고 있다. UML 기반으로 작성되는 소프트웨어 분석 · 설계 산출물에는 유스케이스(Use Case) 다이어그램, 클래스 다이어그램 등이 있으며 UML이 다른 모델링 언어와 구별되는 점은 모델링 대상을 클래스, 활동, 상태 등의 관점에서 구조를 표현하는 다이어그램과 행위를 표현하는 다이어그램으로 구분하여 작성하게 된다는 것이다[9]. UML과 같은 시각적 정보 형태의 모델은 개발자와 고객 또는 개발자 상호 간에 원활한 의사소통을 지원한다.

특정 도메인 또는 대상을 모델링 언어로 추상화하여 모델을 작성하고 작성된 모델을 활용하여 소프트웨어를 개발하는 방법론을 MDE(Model Driven Engineering)라고 한다. MDE의 설계에서 표준 모델의 재사용은 설계 과정을 단순화시키고, 소프트웨어 간의 호환성을 극대화시켜 생산성을 향상시키는 효과가 있다[7]. 하지만 Schmidt[13]는 MDE 플랫폼의 복잡성을 완화시키고 효과적으로 도메인을 모델링하기 위해서는 다른 종류의 모델링 언어와의 연계가 필요하다고 언급하고 있다. 즉, 복합 소프트웨어 또는 대형 소프트웨어를 모델링하기 위해서는 다양한 모델링 언어를 운영하는 능력이 MDE에 요구된다. MDE 모델링 언어로 UML이 주로 사용되지만 UML 뿐만 아니라 다른 종류의 모델링 언어와 함께 사용되는 MDE 확장성 즉, 모델링 언어의 통합이 필요하다.

소프트웨어 공학 분야에서 소프트웨어 확장성은 중요한 이슈로 다루어져 왔고 확장성 문제를 극복하기 위한 많은 해결책들이 제시되었다. 하지만 MDE 확장성 문제는 소프트웨어 공학에서 연구된 자원 공급 확대 방법으로는 해결이 어렵다. 왜냐하면 모델은 다양한 모델 사용자가 협력적으로 개발하고, 대형 모델을 다루기 위한 컴퓨팅 자원의 한계와 모델링에 사용되는 표기법 및 모델링 도구의 다양성 때문에 MDE 확장성 문제는 소프트웨어 공학의 확정성 해결방법과 차이가 있다[15].

MDE 플랫폼의 복잡성을 완화시키고 다양한 종류의 모델링 언어와의 연계하기 위한 방법으로 모델링을 새롭게 하거나 참조하는 방법이 제시되었다. 하지만 이와 같은 방법은 각각의 모델링 언어가 갖는 특징과 장점을 손상시키고 많은 비용과 시간의 투자가 요구되는 모델 통합 방법이다. 그러므로 모델링 언어의 기본 구조를 참조하고 Linked Data 기반으로 모델의 메타데이터를 활용하는 모델 통합을 본 논문에서 제시한다.

본 논문의 구성은 먼저, 제 2장에서 모델 통합의 개념, 형태가 다른 모델을 통합하는 방법, 모델의 개체(Entity)를 중심으로 내부 또는 외부 모델을 통합하는 기존 연구를 고찰한다. 제 3장에서는 모델링 언어의 기본 개념인 메타모델 구성요소, 모델의 메타데이터를 표현하기 위한 Linked Data의 개념, Linked Data 구축 방법을 살펴본다. 제 4장에서 모델의 메타데이터를 Linked Data로 작성하기 위한 메타모델의 구성요소를 RDF 스키마로 정의하고 제 5장에서 서로 다른 그림 형태의 모델, 그림과 서술 형태의 모델에 대해 Linked Data기반의 모델 메타데이터를 활용하여 통합하는 사례를 제시한다. 마지막 제 6장에서는 연구 결과와 향후 과제를 기술한다.

2. 기존 연구

대형 소프트웨어는 복잡하고 복합적인 모델로 분석 · 설계되어 개발된다. 이렇게 작성된 모델에는 수 만 개 이상의 개체(Entity)가 포함된다. 하지만 모델의 개체가 많아지면 모델을 이해하기도 어려울 뿐만 아니라 모델의 개체를 파악하는데 많은 노력이 필요하다. 심지어 일부 모델링 언어는 모델의 개체 개수를 제한하므로 모델의 확장에 문제가 있다. 이러한 모델 확장의 문제를 해결하기 위해 하나의 모델을 여러 부분적인 모델로 나누어

서 모델링한다. 그리고 부분으로 분리된 모델과 모델의 개체를 관리하기 위해 식별 번호를 부여하고 상호간에 참조하여 모델링한다[11].

이와 같은 모델 확장성의 문제를 해결하기 위해 모델 통합 측면에서 연구가 진행되었다. 신규로 작성되는 모델이 기존 동일한 모델링 언어로 작성된 모델에 어떻게 적응되고, 서로 다른 모델링 언어로 작성된 모델을 어떻게 통합을 시킬 것인가를 모델 통합 분야에서 연구되었다. 기존 연구된 네 가지 모델 통합 방법에 대해 살펴보면 다음과 같다.

2.1 전체 통합과 부분 통합

각각의 모델을 단일 모델링 언어로 완전히 새롭게 결합하여 새로운 모델로 통합하는 방법과 각각의 모델에서 관계되는 부분만을 결합하여 단일 모델링 언어로 통합하는 방법이 연구되었다. 전자의 모델 통합 방법을 “Deep” 통합이라 하고 후자의 모델 통합 방법을 “Functional” 통합이라고 한다[6]. 두 모델 통합 방법은 여러 모델링 언어로 작성된 모델을 하나의 모델로 통합하는 것에 많은 시간과 비용을 소요한다.

2.2 중재 통합

모델은 UML과 같이 심벌과 선을 사용하는 그림 형태의 표현 방법이 일반적이다. 하지만 사람과 컴퓨터가 이해할 수 있는 자연어를 사용하여 표현하는 서술 형태의 모델도 작성된다. 그림 형태의 모델링은 개념적으로 이해가 용이하고 서술 형태의 모델링은 의미적으로 이해가 용이하다는 장점이 있다. 하지만 그림 형태의 모델링 방법이 서술 형태의 모델링 방법보다 우수하다고 단정하기 어렵다[12]. 또한 그림과 서술은 형태적인 차이로 직접적인 통합은 불가능하여 중재하는 모델링 언어가 사용되어야 한다[5]. 하지만 모든 모델링 언어를 공통적으로 중재하는 모델링 언어는 존재하지 않는다. 그러므로 각각의 모델링 언어에 따라 중재하는 모델링 언어를 정의하여 모델을 통합한다.

2.3 참조 통합

전체 모델 중에서 일부 필요한 모델만을 사용하거나 관계있는 모델만을 활용하여 통합하는 것이 효과적이기 때문에, 특정 도메인을 대상으로 특화된 모델링 작업을 수행하는 DSML(Domain Specific Modeling Language)이 개발되었다. DSML은 분산되어 있는 모델들을 재통합하고 다른 모델 개발자가 작성한 모델도 재사용할 수 있는 모델링 언어 개념이다. 그림 형태 모델 작성 도구의 기능을 활용하여 모델링 결과를 XML(eXtansible Markup Languag) 기반의 서술 형태 모델로 변환하고, 식별된 모델 객체(Object)를 공통적인 저장소(Repository)에 등록하여 참조한다. 그리고 모델에 사용되는 객체 명칭의 문자열을 중심으로 모델을 통합한다[14]. 이와 같은 모델 통합 방법은 모델에 사용되는 객체 명칭의 문자열에 대한 의미적인 동일 문자열을 고려하지 못하여 모델 통합이 불완전하고 오류가 발생될 수 있다. 또한 모델링에 사용되는 모든 객체를 저장소에 등록하여 재사용하는 방법에서도 재사용되지 않는 대상까지 불필요하게 관리된다.

2.4 모델 통합 방법 종합 비교

앞에서 살펴본 모델 통합 방법을 정리하면 <표 1>과 같다. 전체 모델 통합, 부분 모델 통합, 중재 모델 통합은 특정한 모델링 언어로의 변경이 필요하므로 물리적 모델 통합이라고 호칭하고, 참조 모델 통합은 모델링 언어의 변경이 없이 상호 참조하므로 논리적 모델 통합이라고 한다. 물리적 모델 통합과 논리적 모델 통합은 이분법적으로 분리되는 것이 아니라 병행하여 모델 통합에 사용된다.

각 모델 통합 방법에 대한 문제점을 정리해 보면 다음과 같다. 전체 모델 통합은 비용과 시간이 가장 많이 소요되고 다양한 모델링 언어가 갖고 있는 특징과 장점을 손상시키는 방법이다. 부분 모델 통합 역시 전체 모델 통합과 동일하게 기존 모델링 언어의 특징과 장점을 손상시킨다. 중재

모델 통합은 모델링 언어가 다양한 상황에서 공통적으로 적용 가능한 모델링 언어를 선택하기 어렵다. 참조 모델 통합은 모델 통합 중에서 시간과 비용이 가장 적게 소요되는 방법이지만 공통으로 사용되는 모델 객체를 식별하고 최신화 하는데 지속적인 노력이 요구된다.

〈표 1〉 모델 통합 방법

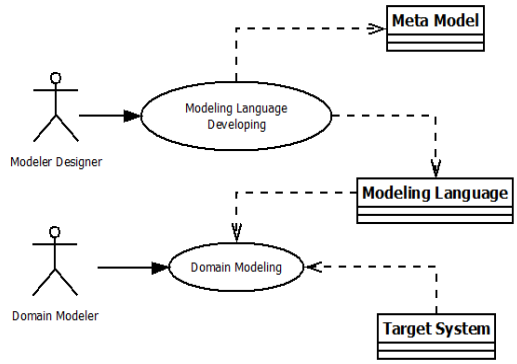
구 분	내용
전체 모델 통합	• 여러 모델링 언어 및 관점으로 작성된 전체 모델을 단일 모델링 언어로 재 모델링하는 물리적 통합
부분 모델 통합	• 여러 모델링 언어 및 관점으로 작성된 모델의 일부만을 단일 모델링언어로 재 모델링하는 물리적 통합
중재 모델 통합	• 여러 모델링 언어 및 관점으로 작성된 모델을 중재하는 모델링 언어로 변경하여 서로 호환시키는 물리적 통합
참조 모델 통합	• 여러 모델링 언어 및 관점으로 작성된 모델의 구성요소가 상호 참조하거나 공통 어휘를 사용하는 논리적 통합

그러므로 다양한 모델링 언어를 사용하여 작성된 모델을 시간과 비용의 노력을 절감하고 통합 대상이 되는 모델의 특징과 장점을 손상시키지 않는 개선된 모델 통합이 요구된다. 이에 대한 해결책으로 모델 자체에 대한 정보, 모델 사이의 관계 정보, 모델과 개체 사이의 정보 구조 등을 제공하는 모델의 메타데이터를 기반으로 모델을 통합하는 방법을 제안한다. 이때, 모델의 메타데이터 구조는 메타모델을 기반으로 Linked data로 정의한다.

3. 관련 연구

3.1 메타모델

모델링 언어는 정보, 지식, 시스템을 대상으로 일관된 규칙 하에 구조적 표현을 제공하는 인공적인 언어이다[8]. 모델링 언어는 특정 도메인에 특화된 모델링 언어로 정의될 뿐만 아니라 UML과 같이 여러 도메인에서 공통적으로 사용 가능한 모델링 언어가 있다. [그림 1]은 실제 모델링되는 시스템 및 소프트웨어를 대상으로 도메인 전문가가 UML 또는 DSML 등의 모델링 언어를 사용하여 추상화시키고, 추상화를 위해 사용되는 모델링 언어가 목적에 따라 메타모델의 구성요소를 조합하여 개발됨을 표현한 것이다.



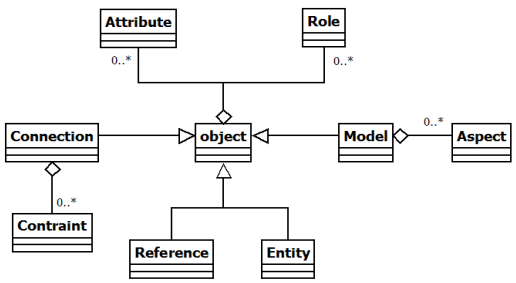
[그림 1] 모델링 과정

Karsai et al.[10]에서, 메타모델은 모델에 기술되는 개체가 모델 내의 다른 개체들과 어떻게 상호 관계를 가지며, 다중적인 관점으로 작성되는 모델에서 개체가 어떻게 연결되며 다른 모델의 개체를 어떻게 참조하는지를 개념화한 것이다. 그리고 모델의 각 개체가 생성되어 운영되기 위한 제약조건을 제시한다. 이러한 메타모델은 대형의 복잡한 소프트웨어를 모델링 하거나 작은 소프트웨어를 모델링하건 간에 모든 모델링 언어에 동일하게 적용된다.

구체적으로 모델링 언어의 메타모델 구성요소를 살펴보면 모델, 개체, 참조, 연결이 있다. 추가적으로 모델 또는 연결은 속성, 역할, 관점, 제약조건의 세부 구성요소가 있다. 메타모델의 각 구성요소에 대한 상세한 설명은 <표 2>와 같다. 이러한 메타모델 구성요소의 상호관계를 UML의 Class Diagram으로 표현하면 [그림 2]와 같다. 이때, 객체지향 개념에서 메타모델의 구성요소는 개별적 객체라고 정의한다.

〈표 2〉 메타모델 구성요소

구성 요소	설명
객체 (Object)	모델, 개체, 참조, 연결, 집합을 대표하는 단위
모델 (Model)	모델링하는 프로젝트에 의해서 작성되는 모델로, 하나의 프로젝트는 요구조건에 따라서 여러 종류의 모델이 작성됨
개체 (Entity)	모델에 포함된 Class 단위의 개체
참조 (Reference)	다른 계층 구조 밑에 있는 객체들을 연결하기 위한 도구로, 다른 프로젝트로 작성된 모델에 있는 개체와 연결하는데 사용
연결 (Connection)	동일한 계층 구조 밑에 있는 객체들 사이의 연결 도구로, 동일한 프로젝트로 작성된 모델에 있는 개체들을 연결
역할 (Role)	객체가 갖는 역할 또는 임무
속성 (Attribute)	객체가 갖는 형태 값 또는 상수 값
관점 (Aspect)	작성되는 모델이 필요한 목적
제약조건 (Constraint)	동일한 프로젝트로 작성된 모델의 개체들이 연결되기 위한 조건



[그림 2] 메타모델 Class Diagram

3.2 Linked Data

정보에 의미를 부여하고 사람뿐만 아니라 기계도 역시 정보에 접근하여 분석할 수 있는 기술을 시멘틱 웹이라고 한다. Linked Data는 데이터 중심의 웹(Web of Data)을 구현하는 시멘틱 웹의 구체적인 기술이다. 문서 중심의 웹에서 데이터 중심의 웹으로 진화되기 위해서는 공통된 언어와 공통된 구조가 요구된다. 공통 어휘(Common Vocabulary)와 메타데이터 모델(Metadata Model)은 공통된 언어와 공통된 구조의 역할을 담당한다. 그리고 공통 어휘와 메타데이터 모델의 표현은 RDF 등의 시멘틱 웹 언어를 사용한다[1]. 특히, RDF 형식은 의미의 모호성을 제거하고 데이터 사이의 변환 및 활용에 유연성을 제공한다.

Linked Data가 구축되는 절차[3]는 6단계이다. 첫 번째 단계에서는 URI(Uniform Resource Identifier) 참조(Reference)를 사용하여 자원을 어떻게 호출할지 명칭을 결정한다. 두 번째 단계에서는 데이터를 표현하기 위해 어떠한 데이터 자원(Source)에서 제공하는 어휘와 용어를 사용할지 선택한다. 세 번째 단계에서는 인터넷에 공개되는 RDF 명세에 어떤 정보를 포함해야 할지 결정한다. 네 번째 단계에서는 다른 데이터 자원에 있는 데이터와 RDF 링크를 설정하는 방법을 결정한다. 다섯 번째 단계에서는 Linked Data를 지원하는 HTTP URIs, RDF/XML 기술뿐만 아니라 RDF 저장소에 Linked Data를 적재하기 위한 데이터 셋의 크기, 데이터 셋을 저장하는 저장소 유형, 데이터 셋의 최신화 주기 등 지원 방법을 정의한다. 마지막으로 구축되어 공개되는 Linked Data가 정상적으로 공유가 되는지를 확인하고 점검한다. 본 논문에서는 모델 통합에 필요한 모델의 메타데이터를 정의한 수준을 연구 대상으로 한다.

4. Linked Data 기반의 모델 통합

4.1 메타모델 온톨로지

RDF 스키마(RDFs)는 RDF를 이용하여 다양한 자원(Resource)을 기술하기 위한 구조를 의미한다. RDF 스키마는 속성에 대한 정의, 속성들 사이의 관계, 자원을 그룹화 하여 클래스로 정의하는 수단을 제공한다[2]. 이러한 RDF 스키마의 기본 클래스에는 rdfs:Resource, rdfs:Class, rdf:Property가 있고, RDF 스키마 특성(Property)을 사용하여 어휘가 정의된다.

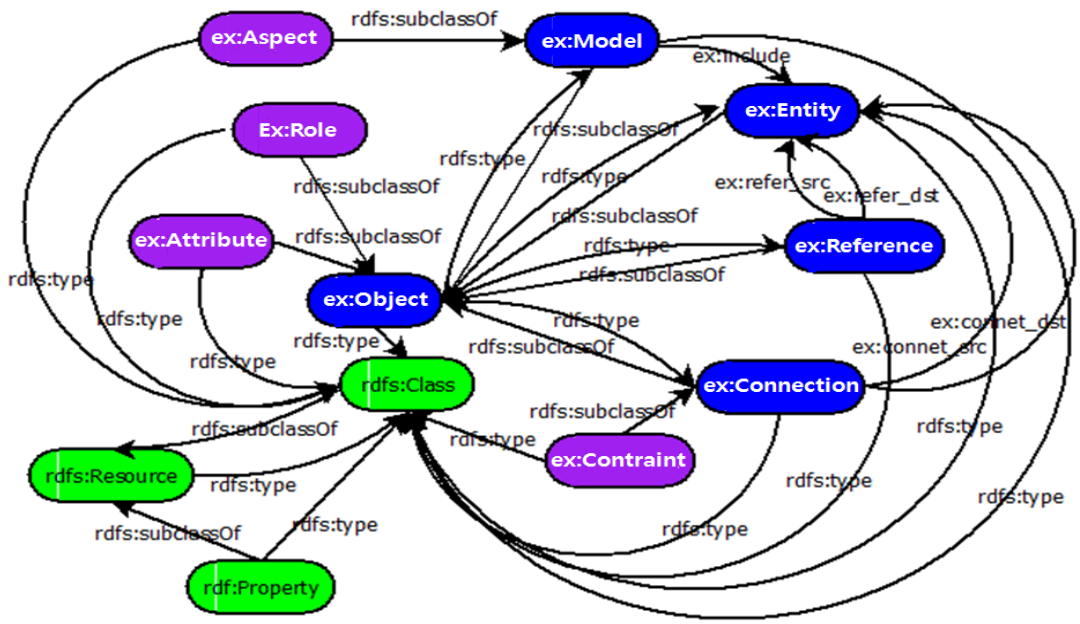
앞에서 언급한 것과 같이 모델에 대한 메타데이

터는 모델링 언어로 작성되는 모델의 객체인 모델, 개체, 연결, 참조 등을 대상으로 Linked Data로 구축한다. Linked Data로 구축되는 모델의 메타데이터는 특정 모델에 포함된 개체 정보뿐만 아니라 관계있는 다른 모델에 포함된 개체 정보까지 포함한다. 이와 같이 Linked Data로 구축되는 모델의 메타데이터를 기반으로 모델을 통합하면 다양한 모델링 언어 및 관점으로 작성된 모델까지 모델링 언어의 특징과 장점의 손실 없이 통합이 가능하다.

모델링 언어의 메타모델을 RDF 스키마 어휘를 사용하여 온톨로지로 표현하며 [그림 3]과 같다. 메타모델의 기본 구성요소는 객체(Object)이고 자원의 서브 클래스이다. 모델링 언어로 작성되는 모델, 개체, 참조, 연결은 객체의 서브 클래스, 속성과 역할은 객체의 서브 클래스, 관점은 모델의 서브 클래스, 제약조건은 연결의 서브 클래스로 정의된다. 모델과 개체, 개체와 연결 또는 참조의 관계 어휘를 RDF 스키마 특성(Property)을 사용하여 정의한다.

4.2 모델의 메타데이터 구축 절차

Linked Data에 의한 모델의 메타데이터 구축 절차는 다음과 같다. 첫 번째, 모델은 특정 목적으로 수행되는 모델링 프로젝트에 의해 생산된 결과이기 때문에 모델 내의 개체를 호출하는 명칭은 모델링 프로젝트 명을 기준으로 URI 참조를 부여하고 호출한다. 두 번째, 모델의 메타데이터에 사용되는 대표적인 어휘와 용어는 RDF 스키마 어휘와 인터넷에서 공용으로 사용되는 더블 코어 용어를 사용한다. 세 번째, Linked Data로 구축되는 모델의 메타데이터는 사용자 효용성을 고려하여 여러 가지 형태로 메타데이터 공개가 가능하고[4], Linked Data의 정보 품질과 신뢰성을 보장하기 위해 데이터 생산자, 생산일자 등의 정보를 추가적으로 메타데이터에 포함한다. 그리고 모델의 메타데이터는 모델링 언어의 메타모델로 포함되는 구성요소 정보를 중심으로 작성한다. 네 번째, 다른 모델링 프로젝트에 의해서 작성되는 모델의 개체를 참조하



[그림 3] RDF 스키마 기반의 메타모델 온톨로지

는 경우에는 “owl : sameAs”를 사용하여 참조한다. 본 논문은 모델 통합을 위해 모델의 메타데이터 구축절차를 제시하는 것으로 Linked Data를 지원하는 기타 기술과 Linked Data를 시험 및 평가 절차에 대한 내용은 제외한다.

모델에 대한 메타데이터가 Linked Data로 구축됨으로써, 그림 형태 또는 서술 형태 등의 다양한 형태로 작성되는 모델도 인터넷상에서 공개하고 공유하여 모델 통합이 가능하다. Linked Data는 RDF 데이터 모델을 기반으로 검색이 요구되는 모델 정보의 모든 URI를 검색 및 추적하여 데이터 스키마에 상관없이 연관되는 모델 정보를 검색하는 장점이 있다. 다음은 모델의 메타데이터를 Linked Data로 표현하기 위한 모델링 언어의 메타모델의 구성요소, 구성요소에 관계되는 인스턴스에 대한 RDF 스키마를 설명한 내용과 표현 사례이다.

4.2.1 객체/속성/역할

메타모델의 주요한 구성요소에는 모델, 개체, 참조, 연결이 있으며 객체(Object)에서 상속된다. 먼저, 객체의 RDF 스키마 클래스는 <표 4>와 같이 정의하고 객체의 속성과 역할은 <표 5>과 같이 객체의 서브 클래스로 정의한다. 객체, 속성, 역할의 인스턴스는 프로젝트 단위로 <표 6>과 같이 프로젝트 구성요소별로 구분하여 별도의 디렉토리에 구축된다.

〈표 4〉 객체 RDF 스키마

```
<rdfs:Class rdf:about =
  “http://example.org/terms/Object”/>
```

〈표 5〉 역할 RDF 스키마

```
<rdfs:Class rdf:about=
“http://example.org/terms/Role”>
      <rdfs:subClassOf rdf:resource=
      “http://example.org/terms/Object”/>
</rdf:Class>
```

〈표 6〉 속성 인스턴스 RDF

```
<ex:Atrribute rdf:about=
“http://example.org/project/Attribute/attribute#attri
bute1”>
      <rdfs:subClassOf rdf:resource=
“http://example.org/project/Object/object#object1”/>
</ex:Atrribute>
```

4.2.2 모델/개체

메타모델의 모델과 개체는 객체의 서브 클래스이다. 모델은 특정한 관점으로 작성되기 때문에 객체의 속성 및 역할과 유사한 개념이고, 관점은 모델의 서브 클래스로 정의된다. 하나의 모델은 여러 개체를 포함하므로 모델과 개체의 포함관계는 RDF 스키마의 기본 특성을 사용하여 <표 7>과 같이 정의된다.

〈표 7〉 모델과 개체의 관계 특성 RDF 스키마

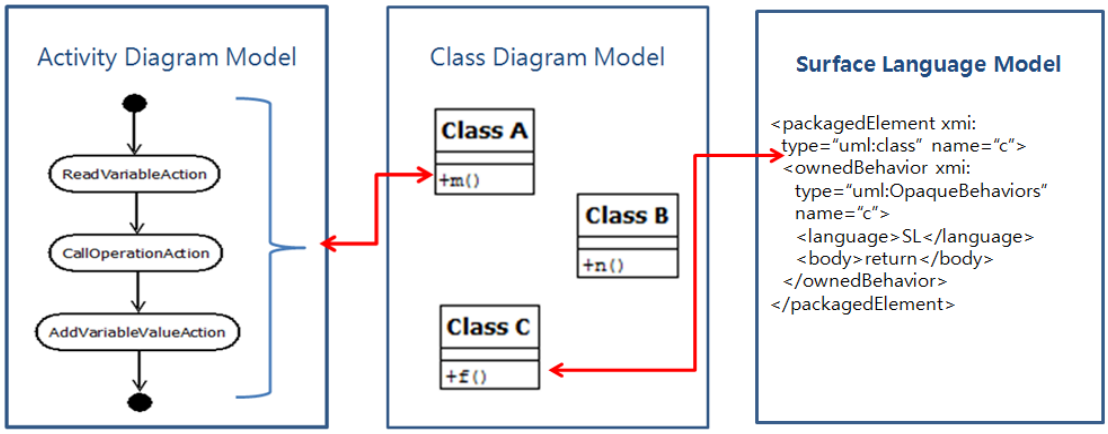
```
<rdf:Property rdf:about=
“http://example.org/terms/Include_enti”/>
```

4.2.3 연결/참조/제약조건

메타모델의 연결과 참조는 모델 및 개체의 수준과 동일하게 객체의 서브 클래스로 정의된다. 제약조건은 모델과 관점의 관계와 유사한 개념으로 연결의 서브 클래스로 정의된다. 이때, 연결과 참조는 출발하는 개체와 도착하는 개체가 정의되어야 하므로 연결과 참조는 RDF 스키마 기본 특성을 사용하여 <표 8>과 같이 정의된다.

〈표 8〉 연결의 출발/목표 개체 RDF 스키마

```
<rdf:Property rdf:about=
“http://example.org/terms/Connect_src”/>
<rdf:Property rdf:about=
“http://example.org/terms/Connect_dst”/>
```



(a) (b) (c)

[그림 4] 모델링 개체 연결 및 참조 사례

5. 사례적용

먼저, 그림 형태로 모델을 작성하는 프로젝트에서 여러 관점으로 작성된 모델의 메타데이터 사이의 모델 통합 사례를 제시한다. [그림 4]의 (a)는 특정한 하나의 클래스에 포함된 오퍼레이션에 대한 순차적인 활동을 표현한 Activity Diagram이고 [그림 4]의 (b)는 클래스 관점으로 전체 클래스들을 표현한 Class Diagram이다. Activity Diagram과 Class Diagram은 동일한 모델링 프로젝트이고 Activity Diagram의 활동과 관계가 있는 Class Diagram의 클래스를 출발 개체와 목표 개체로 하여 <표 9>와 같은 RDF 스키마로 상호 관계됨을 보인다.

다음은 그림 형태로 모델을 작성하는 프로젝트의 모델과 서술 형태로 모델을 작성하는 프로젝트의 모델을 통합하는 사례이다. [그림 4]의 (c)는 다른 모델링 프로젝트에서 작성된 서술 형태의 모델로 [그림 4]의 (b)에 있는 하나의 클래스에 대한 세부 활동을 기술하고 있다. 그림 형태의 모델 개체가 서술 형태의 모델 개체를 "owl:sameAs"를 사용하여 참조됨을 <표 10>과 같이 RDF 스키마로 표현한다.

〈표 9〉 그림 형태 모델의 인스턴스 RDF

```
<ex:Model rdf:about=
"http://example.org/project1/Model/ClassDiagramM
odel">
    <ex:Include_enti rdf:resource=
"http://example.org/project1/Entity/Class A"/>
</ex:Model>
<ex:Model rdf:about=
"http://example.org/project1/Model/ActivityDiagra
mModel">
    <ex:Include_enti rdf:resource=
"http://example.org/project1/Entity/ReadVariableAc
tion"/>
</ex:Model>
<ex:Connection rdf:about=
"http://example.org/project1/Connection/connect#co
nnect1">
    <ex:Connect_src rdf:resource=
"http://example.org/project1/Entity/Class A"/>
</ex:Connection>
<ex:Connection rdf:about=
"http://example.org/project1/Connection/connect#co
nnect1">
    <ex:Connect_dst rdf:resource=
"http://example.org/project1/Entity/ReadVariableAc
tion"/>
</ex:Connection>
```

〈표 10〉 서술형태 모델의 인스턴스 RDF

```
<ex:Model rdf:about=
"http://example.org/project1/Model/ClassDiagramMo
del">
    <ex:Include_enti rdf:resource=
"http://example.org/project1/Entity/Class C"/>
</ex:Model>
<ex:Model rdf:about=
"http://example.org/project2/Model/SurfaceLanguage
Model">
    <ex:Include_enti rdf:resource=
"http://example.org/project2/Entity/packagedElement
"/>
</ex:Model>
<ex:Reference rdf:about=
"http://example.org/project1/Reference/refer#refer2">
    <ex:Refer_src rdf:resource=
"http://example.org/project1/Entity/Class C"/>
</ex:Reference>
<ex:Reference rdf:about=
"http://example.org/project1/Reference/refer#refer2">
    <ex:Refer_dst>
        <owl:sameas rdf:resource=
"http://example.org/project2/Entity/packagedElement
"/>
    </ex:Refer_dst>
</ex:Reference>
```

6. 결 론

모델 통합은 소프트웨어가 대형화되고 복합 시스템이 개발됨에 따라 중요성이 강조되고 있어 많은 연구가 진행됨을 살펴보았다. 하지만 다양한 모델링 언어를 사용하여 작성된 모델을 공통된 모델링 언어 형태로 모델을 변경하거나 공통된 구성요소를 식별하여 공유하는 방법은 각 모델링 언어가 가진 특징과 장점이 손실되는 문제와 구성요소의 최신화 문제가 있다. 이와 같은 문제를 해결하기 위해서 본 논문에서는 모델에 포함된 구성요소 정보를 Linked Data로 모델에 대한 메타데이터로 표현함으로써 모델을 통합하는 방법을 제안하였다. 모델의 개체 및 관계, 외부 모델의 개체 및 관계에 대한 메타모델을 기반으로 메타모델 구성요소와 인스턴스의 관계에 대한 스키마를 제시하였다. 그리고 두 개의 그림 형태 모델간, 그림과 서술 형태 모델간의 Linked Data 기반 메타데이터를 활용하여 모델 통합 사례를 제시하여 제안 방법의 타당성을 증명하였다.

본 논문에서 제시하는 모델 통합 방법은 다양한 형태의 모델을 통합하고 모델을 확장시키기 위한 효과적인 방법이다. Linked Data 기반의 모델 통합 방법을 국방에서 구축된 EA(Enterprise Architecture)에 적용한다면 방대하게 구축된 EA의 모델과 개체의 이해 및 통합에 효과가 있을 것이고 무기체계 소요제기 및 조직·업무 변경의 영향도 등을 파악하는데 기여할 것으로 판단된다.

참 고 문 헌

[1] 김성혁, 도서관 자료공개와 활용을 위한 Linked Data 기술, 국회도서관보, 제48권, 제3호(2011), pp.24-33.

[2] 황석형, 양해술 공역, "시멘틱 웹을 위한 RDF/OWL 입문", 홍릉과학출판사, 2008.

[3] Bizer, C., R. Cyganiak, and T. Heath, How to publish linked data on the web, http://www4.wiwiss.fuberlin.de/bizer/pub/LinkedDataTutorial/, 2007.

[4] Bizer, C., T. Heath, and B.-L., Tim, "Linked Data-The Story So Far", *International Journal on Semantic Web and Information Systems*, Vol.5, No.3(2009), pp.1-22.

[5] Engelen, Luc, and Mark van den Brand, "Integrating textual and graphical modelling languages", *Electronic Notes in Theoretical Computer Science* Vol.253, No.7(2010), pp.105-120.

[6] Gagliardi, Marco, and Cosimo Spera. "Some new results in model integration", *System*

Sciences, 1995. Vol. III. Proceedings of the Twenty-Eighth Hawaii International Conference on. Vol.3. IEEE, 1995.

[7] http://en.wikipedia.org/wiki/Modeldriven engineering.

[8] http://en.wikipedia.org/wiki/Modeling Language.

[9] http://en.wikipedia.org/wiki/Unified Modeling Language.

[10] Karsai, G., M. Maroti, A. Ledeczi, J. Gray, and J. Sztipanovits, Composition and Cloning in Modeling and Meta-Modeling, IEEE Transactions on Control System Technology, 2003.

[11] Kolovos, D. S., Richard F. Paige, and Fiona AC Polack. "The grand challenge of scalability for model driven engineering", *Models in Software Engineering*, Springer Berlin Heidelberg, (2009), 48-53.

[12] Petre, M., "Why looking isn't always seeing: readership skills and graphical programming", Communications of the ACM, Vol. 38(1995), pp.33-44.

[13] Schmidt, D. C., "Guest editor's introduction : Model-driven engineering", *Computer*, Vol.39, No.2(2006), pp.25-31.

[14] Tolvanen, J.-P. and Steven Kelly, "Integrating models with domain-specific modeling languages", *Proceedings of the 10th Workshop on Domain-Specific Modeling*. ACM, 2010.

[15] Van Der Straeten, Ragnhild, Tom Mens, and Stefan Van Baelen. "Challenges in model-driven software engineering", *Models in Software Engineering*, Springer Berlin Heidelberg, (2009), pp.35-47.

◈ 저 자 소 개 ◈

김 대 환 (neohwan@naver.com)

현재 국방대학교 국방관리대학원 박사과정에 재학 중이며, 국방정보체계를 전공하고 있다. 공군사관학교에서에서 이학사, 국방대학교 국방관리대학원에서 전산학 석사 학위를 취득하였고, 주요 관심분야는 EA, NCW, C4I, 체계통합 등이다.

정 찬 기 (ckjung34@gmail.com)

현재 국방대학교 국방관리대학원 국방정보체계학과 교수로 재직 중이며, 공군사관학교에서 공학사, 미국 플로리다공대에서 전산학 석사와 박사 학위를 취득하였고, 주요 관심분야는 EA, 상호운용성, NCW, C4I, 체계통합 등이다.