ISSN 2287-4348
Vol. 5 No. 2

한국스마트미디어학회 & 한국전자거래학회

2016 추계학술대회

장소 : 광주시청자미디어센터 / 호남대학교
일시 : 2016. 10. 28 (금) ~ 29 (토)

PROCEEDINGS
논문집

Smart Media
KOREAN INSTITUTE OF SMART MEDIA
한국스마트미디어학회
한국전자거래학회
Society for e-Business Studies
호남대학교
HONAM UNIVERSITY
(재)전남정보문화산업진흥원
Jeonnam Information & Culture Industry Promotion Agency

논문 발표순서 (Poster Session 2)

P35 283Page
제목 : 라즈베리파이 기반 농업용 스마트 미러 설계
저자 : 신상훈, 김찬중, 이성근(순천대)

P36 286Page
제목 : 아두이노를 이용한 알약 알림 시스템
저자 : 김환희, 류건웅, 이대우, 이성근(순천대)

P37 289Page
제목 : 중소 의료기관 내 악성코드 탐지 특징과 관련 보안위협 현황 확인
저자 : 홍승균, 문창희, 김준영, 정재호((주)에브리존-기술연구소)

P38 293Page
제목 : 변형된 스테이너 시스템을 이용한 분산 저장 부호의 설계
저자 : 카림하룬, 박호성(전남대)

P39 295Page
제목 : 이종 태양광 시스템의 모델 변환을 위한 메타모델에서 자동 트리 모델 생성
저자 : 손현승, 김영철(홍익대)

P40 299Page
제목 : 빅데이터 시각화 방법과 사례분석
저자 : 문희정(호남대)

P41 302Page
제목 : 360도 비디오 서비스 기술을 활용한 개인 적성검사기반 가상직업체험 서비스 플랫폼 구축
저자 : 하태진, 김국정(비온시이노베이터), 조승원, 오홍근(레몬소프트), 박정민(조선이공대)

P042 305Page
제목 : A Study of Hacking Identification and Measurement of the Effect
저자 : Daehwan Ahn, Byungjoon Yoo(서울대)

P43 308Page
제목 : 한의학 침술훈련을 위한 침술 시뮬레이터 구현 방법에 관한 연구
저자 : 민병국, 정희자, 이현준(비온시이노베이터), 박정민(조선이공대), 김남호(호남대)

이종 태양광 시스템의 모델 변환을 위한 메타모델에서 자동 트리 모델 생성

손현승, 김영철
홍익대학교 소프트웨어공학연구실
e-mail : {son, bob}@selab.hongik.ac.kr

An Automatic Tree Model Generation from Metamodel for Model Transformation of Heterogenous Photovoltaic System

Hyun Seung Son, R. Young Chul Kim
SE Lab, Dept. of Computer Information Compunctions, Hongik University

요 약

기존의 다른 정의 언어 XML 스키마, 데이터베이스 스키마, EBNF 등과는 다르게, 메타모델은 모델을 정의할 수 있는 언어로써 객체지향 메커니즘을 사용한다. 객체지향 기법은 속성과 메서드가 결합된 클래스를 재사용할 수 있도록 하여, 메서드 단위의 재사용 보다 쉽고 오버라이딩과 오버로딩을 통한 확장이 용이하다. 그러나 설계의 상속 개념은 노드의 깊이에 따라서, 부모 클래스로부터 상속받은 타입이 무엇인지, 또는 클래스간의 연관성 파악이 어렵다. 모델 변환에서 메타모델의 구조를 파악하지 못할 경우 변환 규칙 작성이 어렵다. 이를 위해, 메타모델을 트리 모델로 자동생성 방법을 제안한다. 이 트리 모델은 모델 변환에서 각 모델의 입력 요소가 되는 메타모델의 구조를 트리로 표현함으로써, 모델 변환시 메타모델 정보의 누락 방지와 메타모델의 구조를 쉽게 파악하고 모델 변환 규칙 작성의 오류를 줄일 수 있다. 현재 이종 태양광 시스템에 적용하고자 한다.

1. 서 론

MDA/MDD(Model Driven Architecture/Model Driven Development) [1-2]는 소프트웨어 개발에서 플랫폼 독립 모델과 종속 모델을 분리하고 독립 모델을 재사용하여 이종 플랫폼을 개발할 수 있는 아이디어를 제공한다. MDA/MDD는 독립모델과 종속모델로 분리하고 두 모델 간의 차이를 모델 변환 기법으로 자동화한다. 그러므로 자동화 도구는 필수이다.

모델 변환 기법[3]은 변환 규칙이나 언어를 통해 모델이나 텍스트를 입력받아 모델이나 텍스트로 자동 생성하는 방법이다. 기존 컴파일러나 프로그램과의 차이점은 변환 규칙이 오픈되어 있어서 컴파일 없이 모델 변환이 수정가능하고 입출력 모델에 대한 메타모델이 제공되기 때문에 모델 구조를 쉽게 파악가능하다.

모델 변환 기법 종류는 모델에서 모델(M2M, Model to Model)[4], 모델에서 텍스트(M2T, Model to Text)[5], 텍스트에서 모델(T2M, Text to Model)[6]이 있다. M2M은 모델에서 모델로 변환하는 방법으로 입력과 출력 모두 모델을 사용하기 때문에 양쪽모델 모두 메타모델이 필요하다. M2T는 모델을 텍스트로 변환하는 방법으로 주로 설계 문서를 프로그램 코드로 변환하는데 사용한다. 이 M2T는 입력되는 모델이 하나이므로 메타모델 하나가 필요하다. T2M은 텍스트에서 모델로 변환하는 방법으로 프로그램 코드를 통해 설계문서를 복원할 때 사용한다. 텍스트를 읽어야 하기 때문에 컴파일러나 파서에서 사용하는 추상구문트리로 변환하고 이를 모델로 변환한다. 이 경우 추상구문트리의 메타모델[7]과 출력되는 모델의 메타모델이 필요하다.

앞에서 살펴본 것과 같이, 모델 변환 기법에서 모델과 모델을 변환시킬 수 있는 규칙만 중요할 것 같지만 입력과 출력을 정의하는 메타모델도 중요하다.

메타모델[8]은 모델을 정의할 수 있는 언어로 다른 정의 언어 XML 스키마, 데이터베이스 스키마, EBNF 등과 같다. 그러나 메타모델은 기존 정의 언어와 다르게 객체지향 기법을 사용한다. 객체지향 기법[9]은 컴퓨터 프로그램을 기능의 목록으로 보는 시각에서 벗어나 객체들의 모임으로 생각하는 것이다. 각 객체는 메시지를 주고받고 데이터를 처리할 수 있다. 이 객체지향 기법은 클래스를 재사용할 수 있도록 하여 메서드 단위의 재사용 보다 쉽고 오버라이딩과 오버로딩을 통해 클래스의 확장이 용이하다.

그러나 설계 문서로 그려진 상속 개념은 노드의 깊이가 깊을수록 부모 클래스로부터 상속받은 타입이 무엇인지 어느 클래스와 연관되어 있는지 파악하기 어렵다. 또한 메타모델의 규모가 커질수록 그 복잡도는 점점 커진다.

모델 변환에서 메타모델의 구조를 파악하지 못할 경우 변환 규칙 작성이 어렵다. 그 이유는 메타모델 구조를 통

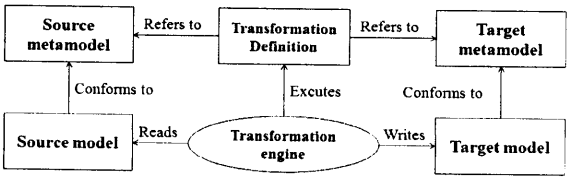
* 이 논문은 2015년 교육부와 한국연구재단의 지역혁신창의인력양성사업의 지원을 받아 수행된 연구임(NRF-2015H1C1A1035548)

해서만 모델의 정보를 가져올 수 있기 때문이다. 이 문제를 위해, 본 논문에서는 메타모델을 트리 모델로 자동생성하는 방법을 제안한다. 이 트리 모델은 모델 변환에서 각 모델의 입력 요소가 되는 메타모델의 구조를 트리로 표현함으로써, 모델 변환시 메타모델 정보가 누락되는 것을 방지하고 메타모델의 구조를 쉽게 파악할 수 있어 모델 변환 규칙 작성의 오류를 줄일 수 있다.

본 논문의 구성은 다음과 같다. 2장에서 모델 변환 기법에 대해 소개한다. 3장에서는 트리 모델 자동 생성 방법을 언급한다. 4장에서는 제안한 내용을 실제 메타모델에 적용하고, 5장에서는 결론 및 향후 연구에 대해 언급한다.

2. 관련 연구

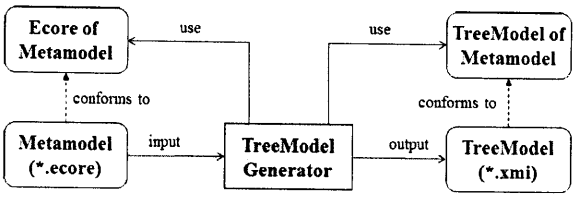
모델에서 모델(M2M, Model to Model) 변환은 모델을 입력하여 모델을 출력하는 방법으로 그림 1과 같은 과정으로 메타모델, 변형 규칙, 변환 엔진으로 구성된다[3]. 모델에서 모델 변환을 위해서는 입력과 출력의 양쪽 메타모델이 정의되어 있어야한다. 여기서 메타모델은 모델정보를 읽어올 수 있는 스키마 역할을 한다. 다음으로 변환 규칙을 정의하는데, 이는 각 모델 요소에 다른 모델 요소로 어떻게 변형되어야 하는지 그 규칙을 기술한다. 즉, 모델의 추가, 삭제, 수정에 관한 내용이 있다. 모델 변환 엔진은 모델 변환 규칙을 실행해 입력 모델을 변환 규칙에 따라 출력 모델로 변환한다.



(그림 1) 모델에서 모델로의 변환 메커니즘

EMF(Eclipse Modeling Framework)[10]는 메타모델을 정의할 수 있는 ecore 모델을 제공하고 모델 변환언어(ATL[11], ETL[12], QVT-O[13] 등) 또는 자바를 사용해 모델 변환을 수행할 수 있도록 환경을 제공한다. 또한 OMG 표준의 데이터 파일인 XMI[14]를 준수하고 있어 모델 변환을 수행하기에 최적의 조건을 가지고 있다.

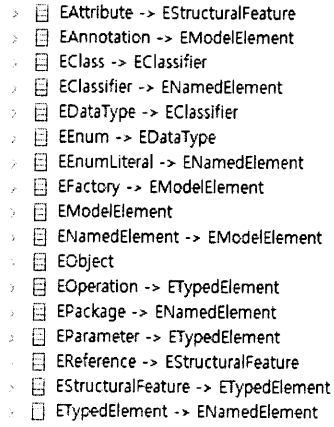
3. 트리모델로 자동생성 방법



(그림 2) 트리 모델 생성기의 구조

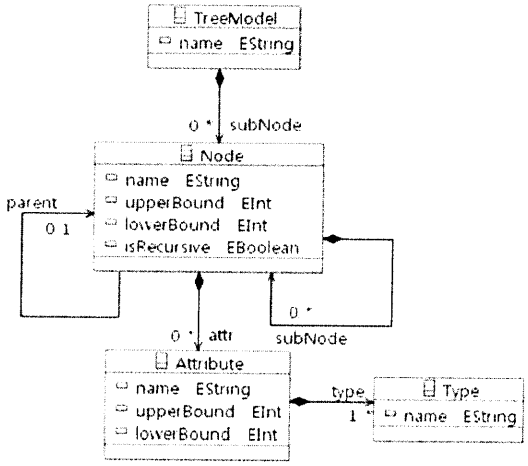
트리 모델 생성기는 그림 2와 같이 메타모델을 입력받고 트리 모델로 변환한다. EMF를 기반으로 메타모델을 처리하기 때문에 Ecore 메타모델을 사용하고 본 논문에서 정의한 트리 모델을 메타모델로 사용한다.

메타모델의 데이터는 그림 3과 같이 상속구조에 대한 모든 정보를 볼 수 있는 것이 아니라 평행적인 구조로 자기 자신의 상위 클래스만 확인할 수 있다. 그러므로 전체 구조를 파악하기 위해서는 일일이 추적해야 하거나 설계 그림을 보면서 찾아야 한다.



(그림 3) ecore의 구조

트리 모델은 메타 모델의 평행적인 구조를 트리 구조로 변환한 것으로 트리 모델의 메타모델은 그림 4와 같다. TreeModel은 최상의 노드로 트리 모델 자체 이름을 저장할 수 있다. TreeModel의 서브 노드는 Node로 메타모델의 부모 노드인 parent와 자식 노드인 subNode를 가지고 있다. 그리고 Node에는 속성과 타입에 해당하는 Attribute와 Type을 가지고 있어 메타모델의 속성과 타입을 표현할 수 있다.



(그림 4) 트리 모델(TreeModel)의 메타모델

트리 모델의 표현은 아래와 같이 다중 관계를 표현하는 [0..*]와 메타모델의 이름, 타입, 속성들로 이루어져 있다.

속성들은 부모 클래스로부터 상속받은 모든 속성이 표현되어 별도의 추적이 없어도 관련된 속성을 한 번에 파악 가능하다.

[0..*] name : Type(attr1:AttrType1 | AttrType2 | ..., ...)

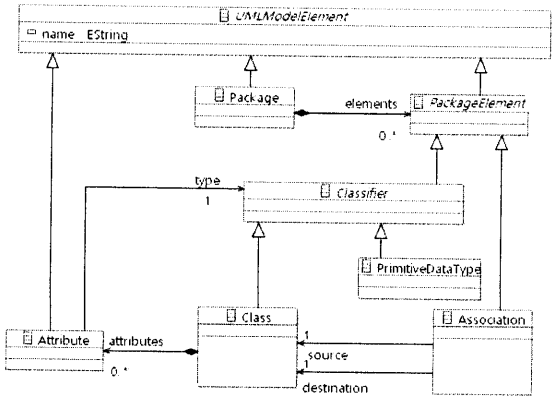
메타모델을 입력 받아 트리 모델을 생성하는 알고리즘의 구조는 아래와 같다. 입력된 메타모델을 너비우선 탐색으로 순회하면서 집합관계이면 서브 노드를 생성하고 생성된 서브 노드에서 추상 클래스일 경우 모든 상위 클래스의 속성과 참조 타입을 가진 노드를 생성한다. 그렇지 않을 경우에는 현재 노드의 속성과 참조 타입을 가진 노드를 생성한다.

1) 입력된 메타모델을 너비우선 탐색을 수행
2) 집합관계이면 서브 노드를 생성
3) 추상 클래스일 경우 : 모든 상위 클래스의 속성과 참조 타입을 가진 노드 생성
4) 그렇지 않을 경우 : 현재 노드의 속성과 참조 타입을 가진 노드 생성

4. 적용사례

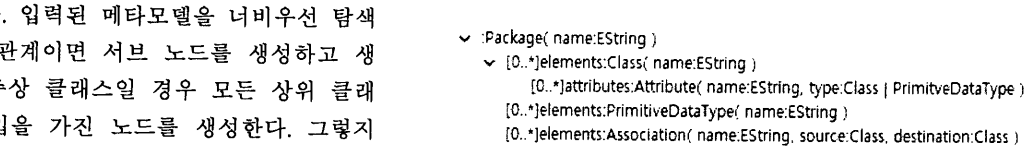
메타모델을 트리 모델로 변환되는 과정을 설명하기 위해 그림 5와 같이 간단한 클래스 다이어그램의 메타모델을 사용한다. 이 메타모델은 Package를 루트 클래스로 여러 개의 Class, PrimitiveDataType, Association 들로 구성되어 있고 Class는 Attribute 들을 가지고 있다.

PrimitiveDataType은 기본적으로 사용할 변수 타입을 지정하고 Association은 클래스와 클래스의 연관관계를 표현할 수 있다. 이렇게 설계된 메타모델은 간단한 구조임에도 불구하고 일반적인 개발자가 그 구조를 쉽게 파악하기 어렵다.



(그림 5) 간단한 클래스 다이어그램의 예

그림 5의 메타모델을 TreeModelGenerator로 트리 모델을 생성하면, 그림 6과 같은 형태의 트리 모델을 만들 수 있다. 생성된 트리 모델을 살펴보면, Package가 루트 노드인 것을 메타모델 설계 도면을 찾아보지 않아도 쉽게 알 수 있다. 메타모델은 루트 노드를 위쪽에 표현하지 않으면 찾기 어렵다. 또한 Package 하부에 Class, PrimitiveData Type, Association이 올 수 있다는 것을 트리 구조로 쉽게 파악할 수 있다.



(그림 6) 간단한 클래스 다이어그램의 트리 모델 변환

5. 결론

메타모델은 모델을 정의할 수 있는 언어로 객체지향 기법을 사용한다. 객체지향 기법은 클래스 단위로 재사용이 가능하고 캡슐화와 확장이 용이하다. 그러나 문서로 표현된 메타모델은 노드의 깊이가 깊을수록 부모 클래스로부터 상속받은 타입이 무엇인지 어느 클래스와 연관되어 있는지 파악하기 어렵다.

이러한 문제는 모델 변환시 메타모델의 구조를 파악하기 어려워 규칙 작성이 어렵다. 본 논문에서는 이 문제를 중점적으로 해결하기 위해 메타모델을 트리 모델로 자동생성 하는 방법을 제안한다. 이 트리 모델은 메타모델의 구조를 트리로 표현함으로써, 메타모델의 구조를 쉽게 파악할 수 있어 모델 변환시 누락되는 정보를 방지가능하다.

현재 이종 태양광 시스템에 적용하고자 트리모델을 이용한 모델 변환 언어를 연구 중이고 이를 확장하고자 한다.

참 고 문 헌

[1] OMG, Technical Guide to Model Driven Architecture: The MDA Guide v1.0.1, Document Number: omg/2003-06-01

[2] S.J. Mellor, A.N. Clark, T. Futagami, "Guest Editors' Introduction: Model-Driven Development", IEEE Software. Vol.20 No.5, 2003. pp.14-18.

[3] K. Czarnecki, S. Helsen, "Feature-based survey of model transformation approaches", IBM Systems Journal. Vol.45 No.3, 2006. pp.621-645.

[4] 김우열, 손현승, 김재승, 김영철, "모델 변환 기법을 활용한 윈도우즈 모바일 어플리케이션 개발", 「정보과학회논문지 : 컴퓨팅의 실제 및 레터」 제16권 제11호, 2011. pp.1091-1095.

[5] OMG, MOF Model to Text Transformation Language, v1.0, OMG Document Number: formal/2008-01-16

[6] J.L.C. Izquierdo, J.S. Cuadrado, J.G. Molina, "Gra2MoL: A domain specific transformation language for bridging

grammarware to modelware in software modernization", MODSE 2008 Workshop On Model-Driven Software Evolution, 2008. pp.1-8.
[7] OMG, Architecture-driven Modernization: Abstract Syntax Tree Metamodel (ASTM) Version 1.0, OMG Document Number: formal/2011-01-05
[8] OMG, OMG Meta Object Facility (MOF) Core Specification Version 2.5, OMG Document Number: formal/2015-06-05
[9] G. Booch, R. A. Maksimchuk, M. W. Engle, B. J. Young, J. Conallen, K. A. Houston, Object-Oriented Analysis and Design with Applications, Addison-Wesley, 1997.
[10] F. Budinsky, E. Merks, D. Steinberg, EMF: Eclipse Modeling Framework, Addison-Wesley, 2009.
[11] Wikipedia, ATL, http://en.wikipedia.org/wiki/ATLAS_Transformation_Language
[12] ETL, http://www.eclipse.org/epsilon/doc/etl/
[13] QVT-O, https://projects.eclipse.org/projects/modeling.mmt.qvt-oml
[14] OMG, XML Metadata Interchange (XMI) Specification Version 2.5.1, OMG Document Number: formal/2015-06-07