MDA(Model Driven Architecture) 기반의 통합 아키텍처 연구 사례

[1]박영식, [2*]김영철, [3]강윤희, [4]박보경, [5]장우성
[1] 홍익대학교 소프트웨어 공학 연구실, 박사과정, park12160422@selab.hongik.ac.kr
[2*] 홍익대학교 소프트웨어 공학 연구실, 교수, bob@hongik.ac.kr
[3] 백석대학교 ICT 학부, 교수, yhkang@bu.ac.kr
[4,5] 홍익대학교 소프트웨어 공학 연구실, 박사과정, {park,jang}@selab.hongik.ac.kr

Research practices on Integrated Architecture based on Model Driven Architecture (MDA)

[1]Park Young Sik, [2*]R. Young Chul Kim, [3] Y. H. Kang and [4]Bo Kyung Park, [5]Jang Woo Sung
[1] SE Lab., Hongik University, Ph.d Candidate, park12160422@selab.hongik.ac.kr
[2*] SE Lab. Hongik University, Professor, bob@hongik.ac.kr
[3] ICT Division. Baekseok University, Professor, yhkang@bu.ac.kr
[4,5] SE Lab., Hongik University, , Ph.d Candidate, {park,jang}@selab.hongik.ac.kr

요 약

최근 국내외적으로 플랫폼의 개발이 오픈사이언스를 위하여 이루어지며,과학 기술 용 데이터 처리 및 서비스 운영을 관찰하여 지식인프라의 성과를 확산시키기 위한 시스템 개발의 요구가 제고되고 있다. 이를 위해 주요 과학기술인프라를 위한 서비스 제공과 데이터처리의 융합서비스를 지원하는 S/W아키텍처의 디자인 및 구축을 위한 주요 지표 확보가 필수적이다.과학 기술 용 지식인프라의 데이터 처리와 서비스의 운영 확대에 따른 효율적 시스템 관리를 위한 통합서비스 모델 확정이 요구된다.

본 연구에서는MDA(Model Driven Architecture)기법을 통하여 현 시스템의 서비스와 컴포넌트를 활용한 새로운 통합서비스를 효율적으로 개발할 수 있는 방법을 제안한다.

Abstract

Recently, platform development for open science has been made at home and abroad, and the demand for system development for performance diffusion of science and technology knowledge infrastructure is increasing by analyzing data processing and service operation of science and technology knowledge infrastructure. To this end, it is essential to prepare the main guidelines for the design and development of software architecture supporting the convergence service of major science and technology infrastructure services and data. In this study, we provide methods for effectively developing new integrated services by utilizing the services and components of existing systems using the method of Model Driven Architecture (MDA).

***Keywords**: Model Driven Architecture (MDA), Service Architecture, Knowledge Infrastructure, Software Architecture, New Integrated Services*

* Corresponding Author
Received: Sept. 13, 2019, Revised: Dec. 02, 2019, Accepted: Dec. 31, 2019

홍익대학교 세종캠퍼스 | IP:203.249.87.*** | Accessed 2020/11/23 17:40(KST)

I. 서론

MDA 는 구동되는 S/W 가 특정인프라의 구조나 플랫폼에 제한을 받지 않는 개발을 수행할 수 있도록 UML 을 근거로 하는 독립 플랫폼의 모델을 제작하고 그 모델을 필요로 하는 플랫폼에 적용할 수 있게 변환하는 차세대 소프트웨어개발 기술이다. MDA 는 OMA(Object Management Architecture)가 그 시초이다. OMG 에서는 1989 년 OMA(ObjectManagementArchitecture)을 발표하였으며,COBRA 가 OMA 에서 가장 중요한 역할을 한다. IDL(Interface DefInition Language)이 CORBA 에서는 다양한 프로그래밍언어의 상호 운용성을 위한 기능이며,독립적인 프로토콜을 위해 IIOP(Internet Inter-Operable Protocol)와 동일한 프로토콜 표준안을 제안한다.

본 논문에서는 MDA 기법을 통하여 특정 인프라스트럭처나 플랫폼에 제한받지 않고 개발할 수 있도록 UML 을 기초으로 플랫폼의 독립적인 모델을 만들고,이를 필요로 하는 플랫폼에 적용되도록 변환하는 방법을 제안한다.본 논문은 다음과 같이 구성된다. 2 장에서는 관련 연구로 MDA 에 대하여 설명한다. 3 장에서는 MDA 기반 통합서비스 적용방안에 대해서 제안한다. 4 장은 결론 및 향후 연구에 대한 내용이다.

II. 관련 연구

2.1 MDA(Model Driven Architecture)

MDA 은 시스템 설계에 있어 각 전문가 관점별로 나누어진 비즈니스,설계,구현모델로 분리하는 Separation of Concern 개념을 가지고 있다. 또한 컴포넌트 기술의 모든 항목에 표준 메타 모델을 정의하고,이를 기초로 구성요소를 정의한다.정의되는 결과는 PIM(Platform Independent Model), PSM(PlatformSpecific Model), Code 로 나누어진다. PIM 은 UML 을 기초로 하는 독립적 플랫폼 모델을 말한다.PSM 은 특정 플랫폼에서만 활용될 수 있는 플랫폼의존형 모델을 말한다. Code 는 C, Java 등의 프로그래밍 언어이다.PIM 은 구현 기술의 추상화 수준이 높은 독립적 모델이고, PSM 은 PIM 에서 변환 규칙을 이용하여 자동적으로 변환시키는 툴의 구현 기술을 적용하는 모델이다. 그로 인하여 PIM 에서의 보완으로 유지보수가 가능하고,그 모델에서 코드를 직접 생성하는 것이 코딩의 시간적 비용을 감소시키고,일관성을 갖춘 결과물을 얻을 수 있다.

2.2 MDA 의 설계목표

-.도메인 전문가들이 완성한 표준 모델들을 제공받아 개발을 위한 최대한의 플랫폼을 준비함으로써 각 개발에 필요한 최대한의 생산성을 목표로 한다.
-. MDA 에서는 CORBA 뿐만 아니라 J2EE, 닷넷, 웹서비스 등의 플랫폼을 포함시켰다.
-. OMA 에서의 개발프로세스와 유사한 방법으로 개발한다.
-. MDA 는 산업도메인 모델을 기반으로 필요한 모델들을 제작한 후, 목표로 하는 최적화 부분을 적정화한다.
-.도메인 모델에 포함되지 않은 모델은 추가 모델링하여 PIM 이 완성된다.
-.MDA 프로세스에 준하여 PSM 으로 Model Transform 하고 해당 플랫폼의 소스코드를 제작한다.

<표 1>는 MDA 의 구성요소를 MOF, UML, XML, CWM 으로 나눈다.

MDA 적용에서는 여섯 가지의 장점을 엿볼 수 있다. 첫 번째, MDA 에서 PIM 과 PSM 을 분리함으로써 PIM 의 변경없이 플랫폼의 변경이나 요구사항에 신속히 대응할 수 있다. 플랫폼에 대한 신규 애플리케이션을 공급하는 경우 PIM 의 변경이 필요하지 않으므로 PIM 을 이용해서 신규 플랫폼에 대한 PSM 을 자동으로 제작한다. 이를 일부 수정하여 코드를 재생성하

홍익대학교 세종캠퍼스 | IP:203.249.87.*** | Accessed 2020/11/23 17:40(KST)

Table 1. The components of MDA

항목	내용
MOF (Meta Object Facility)	모델 정보에 대한 표준 저장공간를 제공하여 표준화된 방식으로 모델 정보를 접근하는 조직를 정의
UML (Unified Modeling Language)	객체와 컴포넌트 시스템을 표현하는 언어
XML (XML Metadata Interchange)	UML 로 기술된 모델 정보의 XML 의 표준
CWM (Common Warehouse Metamodel)	데이터 공간 통합에 대한 표준을 정의하고 DB 모델과 스키마 변화 모델, OLAP, Data-mining 모델에 대한 표준화 방법을 제공

는 것으로 새로운 기술 플랫폼에 대한 대응이 용이하다. 두 번째, 플랫폼의존형 시스템은 현재의 아키텍처가 새로운 요구사항에 대응하지 못하는 경우가 많다. MDA 를 활용하면 기능과 아키텍처의 분리정의로 인하여 아키텍처의 변화가 있는 경우 변환 및 구현을 용이하게 할 수 있기에 기능과 요구사항 변경으로부터 아키텍처 변경이 어렵지 않다 .이는 시스템의 생명주기를 연장시키고,나아가서는 안정된 시스템 유지를 가능케 한다. 세 번째, MDA 는 모델의 자동화와 변환이 다양한 플랫폼을 원활히 지원하고 많은 시간이 요구되는 코드 작성 줄여서 보다 쉽게 유지보수가 가능하며,신속한 버그 체크가 가능하다 .이미 제작된 PIM 은 비즈니스의 핵심모델로 향후 상이한 시스템에서 쉽게 도입될 수 있고, 재사용성의 수준 또한 향상된다.네 번째, MDA 에서는 모델과 코드, 문서가 항시 동기됨으로써 문서 작업에 필요한 일의 공수를 줄인다. 다섯 번째, MDA 모델 자동화와 테스트 툴을 이용할시 개발자들이 하는 테스트를 애플리케이션 모델 수준에서 가능하기에 디자인과 애플리케이션 로직의 미비점을 단시간에 포착한다. 이는 품질관리에 투입되는 COST 도 감소한다. 여섯 번째는, PIM 의 간결함은 질높은 시스템 구축을 가능케 하며,모델링은 팀원간의 원활한 의사소통이 가능하게 한다. 또한 결점이 발견되면 신속히 해결하도록 지원한다. 잘 정리된 코딩 패턴을 모델에 적용하는 MDA 의 자동화 도구들은 일반적으로 작성되는 코드에 비하여 결손율이 낮다.

III. MDA 기반 통합 서비스 적용 방안

3.1 Metamodel

메타모델링은 모델의 정보를 저장한 템플릿화를 말한다.이는 모델이 데이터를 표시하는 정보를 가지고 있는 경우, 메타모델이 모델을 나타내는 정보를 가지고 있다는 의미이다.

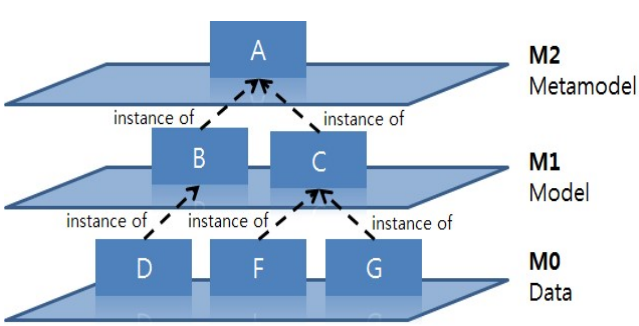

Figure 1. The metamodel hierarchy

홍익대학교 세종캠퍼스 | IP:203.249.87.*** | Accessed 2020/11/23 17:40(KST)

<그림 1>는 메타모델의 계층을 표시한다. M0 은 데이터 계층, M1 은 모델 계층, M2 는 메타모델 계층을 표시한다. M0 단계의 데이터는 M1 단계의 모델을 상속으로 생성되고, M1 단계의 모델은 M2 단계의 메타모델을 상속받아 생성된다. A 모델을 이용한 B, C 모델들을 정의하였다면, B, C 모델들은 A 모델을 이용한 접근이 가능하다. 즉, A 모델의 구조를 알면 B, C 모델에 의해 생성된 데이터를 사용할 수 있다.

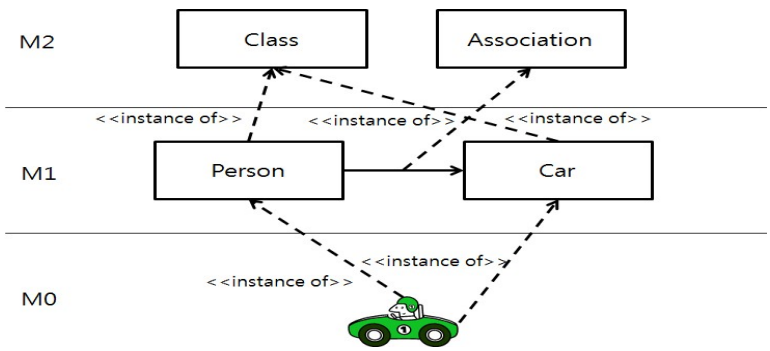

Figure 2. The example of metamodeling

<그림 2>은 메타모델링의 예이다. 여기서는 데이터를 UML 의 Class Diagram 으로 모델링하고 Class Diagram 을 메타모델링하는 과정을 나타낸다. 메타모델링으로 하기 위해서는 M0 에서 데이터를 구성하는 요소를 추출한 후 M1 단계의 모델로 정의한다. M0 단계는 자동차를 타고 있는 그림이다. M0 단계에서는 Person 및 Car 를 모델로 추출한다. 모델의 메타모델을 만들기 위하여 모델의 요소를 추출하고 Class Diagram 으로 정의하여야한다. M1 단계는 Person 및 Car 의 클래스가 존재하고, Person 및 Car 클래스가 연관성을 갖고 있다. 즉, M1 단계의 모델은 Class,Association 두 가지 요소를 갖는다. M2 단계에서는 이 두 요소를 추출,메타모델로 정의한다.

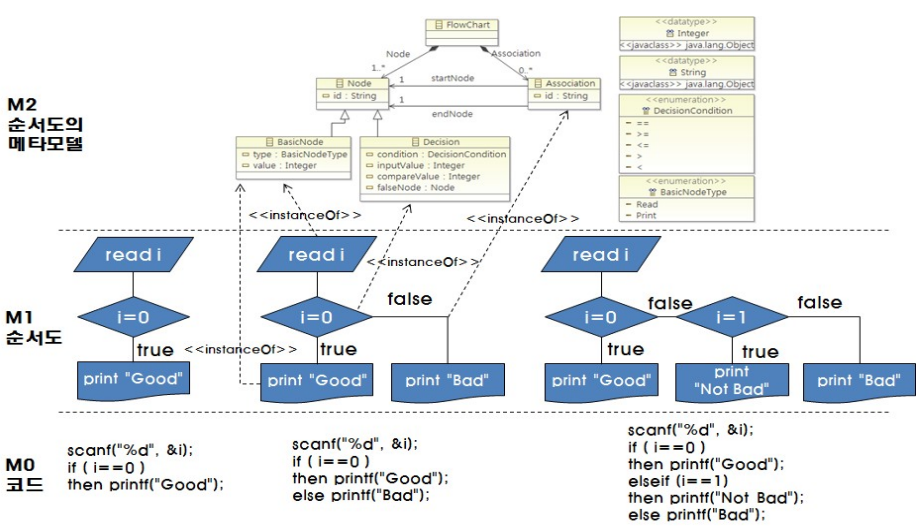

Figure 3. The example of a flowchart model expressed as a metamodel

<그림 3>는 if-then, if-else, if-elseif 들의 소스코드를 근거로 하여 순서도 모델을 정의한 후, 그 순서도 모델을 기반으로 메타모델을 정의한다. M0 단계는 데이터이며,실제 C 언어 코드로 작성되었다. 좌측부터 if-then, if-else, if-elseif 구조이다. M1 단계에서는 데이터 모델을 정의한다. C 언어 코드의 순서도를 이용하여 모델을 정의하였다. M2 단계는 순서도의 메타모델을

홍익대학교 세종캠퍼스 | IP:203.249.87.*** | Accessed 2020/11/23 17:40(KST)

정의하고, 메타모델 정의를 위해 순서도의 구성요소를 먼저 파악한다. 순서도의 구성요소는 Node 와 Association 으로 나눈다. Node 는 정보를 포함하는 요소이다. Node 는 Read, Print, Decision 이 존재한다. Read 는 C 언어 코드의 scanf 함수와 매칭되고, Print 는 코드의 printf 함수와 매칭된다. Decision 은 if-else-then 와 매칭되고 있다. 여기에서는 Read 와 Print 를 동일한 것으로 판단하여 BasicNode 로 정의하였기에 Nod 는 BasicNode 와 Decisio 의 2 종류로 나뉜다. UML 의 Class Diagram 으로 표현하는 경우, BasicNode 및 Decision 의 클래스는 Node 클래스를 상속받는다. Association 은 Node 를 링크하는 정보를 보유한다. 즉 시작 Node 및 마지막 Node 의 정보를 알려준다. Class Diagram 으로 표현하면, Association 은 Node 클래스와 연관관계를 가지는데, 최상위 클래스의 FlowChart 클래스는 Node, Association 클래스를 구성요소로 갖고, Node 클래스의 종류는 BasicNode 와 Decision 클래스로 구분된다. Association 클래스는 Node 클래스를 상속받는 BasicNode, Decision 클래스와 연관관계를 갖는다.결론은 M1 단계의 순서도 모델은 세 가지이지만, M2 단계의 메타모델 구조만 알아도 세 모델에 모두 접근이 가능하다.

3.2 MOF(Meta Object Facility)

OMG 에서 제안한 메타 객체 관리 기법으로 표준에 근거한 메타모델을 정의할 때는 MOF 가 사용된다. MOF 는 UML 을 통하여 표현되고, UML 의 메타모델은 MOF 를 통하여 정의된다. MOF 가 활용되면 모델의 의미에 일관성을 부여할 수 있고, 모델간의 정보 교환을 용이하게 하며, 표준에 준하는 다양한 정보 제공이 가능하다. "OakimIsaksson"의 논문에서 MOF 를 기반으로 하여 메타모델을 작성하는 경우 객체지향적인 다이어그램으로 메타모델을 정의할 수 있다고 한다. 이것은 MOF 를 사용하여 객체지향 설계 기반의 다이어그램은 표준화된 방식으로 메타모델 설계가 가능하기에 다양한 모델의 의미를 일률적으로 통일할 수 있다. “Atthew J.Emerson”의 논문에서는 모델간 변형에 사용되는 모델을 메타모델이라 하고, "Francisco Ruiz"의 논문에서는 MOF 기반의 메타모델은 XMI 를 통해 정보를 교환할 수 있어 다른 Tool 에 의하여 만들어진 모델은 이식이 가능하다고 한다. 메타모델을 이용하면 모델 간의 변형 정보를 유지할 수 있으며, MOF 에서는 모델 간의 정보교환에 있어서는 XMI 문서를 제공함으로 MOF 를 사용할시 정보교환이 용이하다. MOF 는 4 개의 계층 구조를 갖는다. 1 계층은 M0(System) 로 우리가 흔히 기술하는 데이터로 구성된다. 2 계층은 M1(Model) 이며, M0 계층의 데이터를 서술하는 메타데이터 모델로 구성된다. 3 계층은 M2(Model of a Model) 로 M1 단계 모델을 정의하는 메타모델로 이루어진다. M2 단계의 메타모델은 M1 단계의 모델의 구조와 의미를 갖는다. 4 계층은 M3(Model of a Model of a Model)로써 M2 단계의 메타모델의 구조와 의미를 가지는 메타메타모델로 구성된다. MOF 에 대한 메타모델은 MOF 를 재차 이용하여 정의할 수 있기 때문에 MOF 구조에서 더 이상의 수준은 필요하지 않다.

<그림 4>는 4 계층 메타모델 간의 구조를 표시하고 있다. M3 단계에서 MOF Class 라는 이름의 클래스가 정의되면, M2 단계에서는 MOF Class 의 상속으로 여러종류의 클래스를 정의한다. <그림 4>에서는 단순하게 표현되지만, MOF 는 메타모델을 정의하기 위한 여러 형태의 메타메타모델을 갖고 있다. 그런 근거로 MOF 를 사용하여 다양한 형태의 메타모델을 정의할 수 있다. M2 단계는 UML Class 및 UML Attribute 클래스를 정의한다. 이들 클래스는 MOF Class 의 상속으로 정의되기에, MOF Class 만 알면 접근과 이용이 가능하다. M1 단계에서는 M2 단계 메타모델의 상속으로 모델을 정의하였다. M2 와 같이 M1 단계의 모델은 M2 단계의 메타모델을 통해 접근이 가능하다. M0 단계에서는 M2 단계 모델의 상속으로 실제 클래스를 정의한다.

IV. 결론 및 향후 연구

MDA 의 장점을 기반으로 이종의 상호운영이 가능하고 코드 및 유지보수 시간의 절약을 유도하여 빠른 개발도 가능하다. 또한 PIM 의 재사용률 제고에 대한 연구가 필요하다. 그를 통하여 결점의 최소화를 추구하며, 품질관리 COST 의 감소를 달성할 수 있을 것으로 본다.

홍익대학교 세종캠퍼스 | IP:203.249.87.*** | Accessed 2020/11/23 17:40(KST)

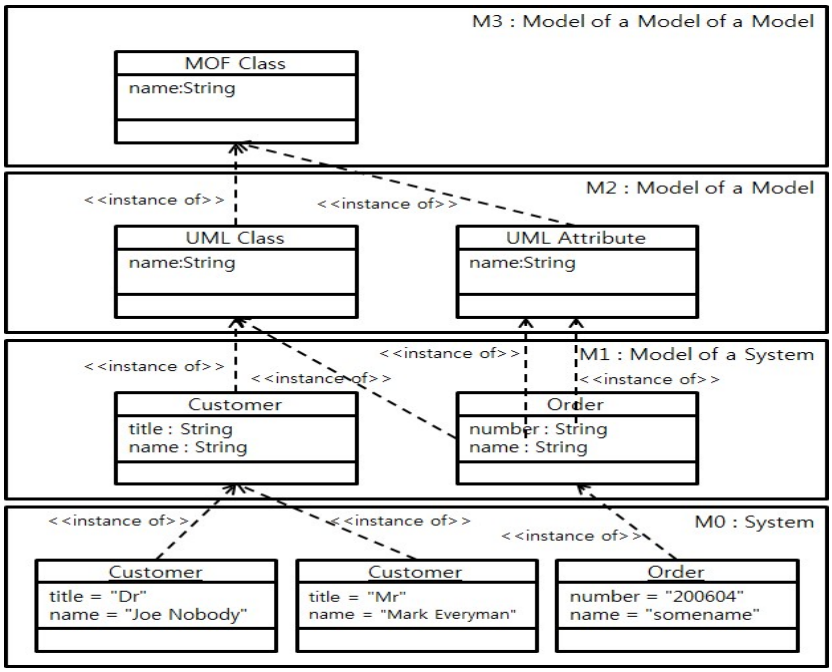

Figure 4. The structure between 4-tier metamodels

V. 감사의 글

본 논문은 한국과학기술정보연구원이 지원하는 2018 년도 위탁연구과제(No. P18014)와 2017 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2017R1D1A3B03035421)의 연구수행으로 인한 결과물 임을 밝힙니다.

VI. 참고문헌

[1] Oakim Isaksson,"A MOF-based metamodel for SA/RT",RISE 2004: Rapid Integration of Software Engineering Techniques,pp.102-111,2004
[2] Hyun Seung Son , Chae Yun Seo , R. Young Chul Kim, "Model Transformation Rule for generating Automatic Database Schema of Business Process Framework Matthew J.Emerson,janos Sztipanovits," International Journal of software Engineering and Its Application, Vol.8 No 3,pp.47-54,2014
[3] Matthew J.Emerson,janos Sztipanovits,"Implementing a MOF Based Metamodeling Environment Using Graph Transformation", Proceeding of OOPSLA Workshop on Domain-Specific Modeling,pp.83-92,2004
[4] Object Management Group, "OMG Meta Object Facility(MOF) Core Specification",2016
[5] Hyun Seung Son, Woo Yeol Kim, Robert YoungChul Kim, Hang-Gi Min," Metamodel Design for Model Transformation from Simulink to ECML in Cyber Physical Systems", Computer Applications for Graphics, Grid Computing, and Industrial Environment,CCIS 351,pp.56-60,2012
[6] OMG,"MOF Model to Text Transformation Language, Vol.1.0,2008

홍익대학교 세종캠퍼스 | IP:203.249.87.*** | Accessed 2020/11/23 17:40(KST)