Journal of The Korea Society of Computer and Information
Vol. 21 No. 12, pp. 115-124, December 2016
www.ksci.re.kr
https://doi.org/10.9708/jksci.2016.21.12.115

Design Automation for Enterprise System based on .NET with Extended UML Profile Mechanism

Deuk-Kyu Kum*

Abstract

In this paper, a method to generate the extended model automatically on the critical elements in enterprise system based real time distributed architecture as well as the platform specific model(PSM) for Microsoft(MS) .NET platform is proposed. The key ideas of this method are real time distributed architecture should performed with satisfying strict constraints on life cycle of object and response time such as synchronization, transaction and so on, and .NET platform is able to implement functionalities including before mentioned by only specifying Attribute Code and maximizing advantages of MDA. In order to realize the ideas, functionalities which should be considered enterprise system development are specified and these are to be defined in Meta Model and extended UML profile. In addition, after definition of UML profile for .NET specification, by developing and applying these into plug-in of open source MDA tool, and extended models are generated automatically through this tool. Accordingly, by using proposed specification technology, the profile and tools easily and quickly reusable extended model can be generated even though low level of detailed information for functionalities which is considered in .NET platform and real time distributed architecture. In addition, because proposed profile is MOF which is basis of standard extended and applied, UML and MDA tools which observed MOF is reusable.

▸ Keyword : MDA, UML Profile, Microsoft .NET, Real-time Distributed Architecture, Attribute code

I. Introduction

최근의 복잡한 비즈니스 환경은 기업의 유연하고 민첩한 대응을 요구하게 되었고, 이를 위하여 컴포넌트 기반 실시간 분산 아키텍처가 엔터프라이즈 시스템과 솔루션들을 지원하는 아키텍처로 널리 받아들여지고 있다 [1]. 이러한 아키텍처에서 객체의 생명 주기와 반응 시간에 대한 객체 풀링, 동기화 서비스, 트랙잭션 서비스 등의 시간 제약성과 보안 서비스 등의 기능은 필수적이다. OMG의 모델기반 아키텍처(Model Driven Architecture: MDA)는 소프트웨어를 개발할 때 재사용의 단위를 소프트웨어 모델로 확장하여 소프트웨어 개발의 효율성을 높이고자 하는 개발 방식이다. 이러한 MDA의 초기 연구는 주로 다양한 변환 포맷 지원이나 여러 개발 언어를지원하는 소스코드 생성기능 등에 초점을 맞추어 진행되었으며 최근에는 기본 베이스 모델로부터 모델 확장에 필요한 요소를 적용하여 확장 모델을 생성할 수 있는 기법에 대한 연구가 진행되고 있다 [2]. 하지만 임베디드 소프트웨어와 같은 특정 분야나 특정 도구에 한정되어 있으며, 앞서 언급한 시간 제약성 등에 대한 모델 확장성 부분은 부족 하였다.

마이크로소프트의 닷넷 플랫폼은 실시간 분산 아키텍처를 지원하며, 코드상에서 애트리뷰트 코드를 명시하는 것 만으로 시간 제약사항, 보안 등의 서비스를 지원하는 컴포넌트를 구현할 수 있어 닷넷 애트리뷰트 코드와 UML 프로파일 메커니즘을 이용한다면 MDA의 장점을 극대화할 수 있다. 하지만, Enterprise Java Beans (EJB) 기술을 위한 메타모델과 UML 프로파일이 발표 [3]된 것에 비해 닷넷은 아직 제정되어 있지

• First Author: Deuk-Kyu Kum, Corresponding Author: Deuk-Kyu Kum
*Deuk-Kyu Kum(dkkum@hanmail.net), Dept. of Computer Software, Dong Seoul University
• Received: 2016. 12. 08, Revised: 2016. 12. 16, Accepted: 2016. 12. 27.

않으며, 닷넷용 플랫폼 종속적 모델(PSM) 명세 기법의 연구가 부족하다.

본 논문에서는 엔터프라이즈 시스템 개발에서 고려해야 할 다양한 서비스의 기능 요소를 명세화하여, 이를 메타모델과 확장된 UML 프로파일로 정의한다. 또한, 닷넷용 PSM 명세를 위한 UML 프로파일을 정의한 후 이들을 오픈소스 MDA 플랫폼인 “StarUML [4]“의 플러그인으로 개발 및 적용함으로써 자동으로 설계 모델을 생성할 수 있도록 한다. 따라서 제안된 명세 기법, 프로파일, 도구를 이용하면 닷넷 플랫폼과 실시간 분산 아키텍처에서 고려할 기능에 대한 하위 레벨의 정보를 자세히 알지 못하더라도 모델에 속해 있는 각 요소들의 역할을 정의하여, 역할에 따른 속성값을 설정해 주기만 하면 쉽고, 빠르게 재사용 가능한 확장 모델을 생성할 수 있다. 제안한 프로파일은 OMG의 UML 프로파일 및 MDA 표준규약의 기반인 MOF를 확장 적용한 것이므로 MOF를 준수한 여러 UML 및 MDA 도구에서 재사용이 가능 하여 설계 모델의 생산성, 확장성, 이식성 및 유지보수성을 증가 시킬 수 있다.

2장에서는 UML 프로파일과 닷넷 애트리뷰트 코드를 소개하고 관련연구를 서술하며, 3장에서는 실시간 분산 아키텍처 기반 서비스 요소 및 이의 메타 모델을 정의한다. 4장에서는 명세한 서비스를 적용한 UML 프로파일을 정의하고, 정의한 프로파일을 이용한 모델 설계 과정, 표현법 및 변환 과정을 설명한다. 5장에서는 사례연구를 기술하고, 6장은 본 논문을 평가하고, 마지막 7장은 결론을 내린다.

II. Preliminaries

1. Related works

1.1 UML 프로파일 (Profile)

UML은 매우 일반적인 언어이기 때문에, 특정한 프로그래밍 언어의 개념을 표현하거나 혹은 특정한 애플리케이션 도메인의 개념을 표현하기에는 부족하다. 이런 경우 확장 속성을 사용할 수 있는데, 이것을 체계적인 형태로 정의해서 하나의 패키지로 만든 것이 UML 프로파일이다. UML 프로파일은 특정 요소를 여러 가지 관점에서 확장할 수 있도록 해 준다. 스테레오 타입을 통해 요소들을 분류할 수 있는 기준을 제공하고, 확장 속성 혹은 꼬리표 값을 통해 요소에 정의되지 않은 또 다른 속성을 정의할 수 있도록 도와준다. 제약사항은 Object Constraints Language (OCL)로 표현될 수 있으며, 시스템의 제약적인 내용을 서술한다 [5]. 이름과 타입을 갖는 꼬리표값은 특정 스테레오 타입에 첨부되어 부가적인 속성값을 표현한다. 이처럼 UML 프로파일은 플랫폼 종속적인 정보를 표현하기 위한 명세 방법과 코드 생성시 필요한 추가적인 상세 설계 정보의 명세방법을 제공한다. 하지만 MDA에서 UML 프로파일은 해당 기술 플랫폼을 표현하기 위한 PSM 수준의 도구로만 사용되는 것은 아니다.

MDA는 그림 1과 같이 PIM 수준에서 특정 도메인에 일반적인 요소들, 예를 들면 컴포넌트 아키텍처에서의 데이터 공유, 이벤트 관리, 프로세싱, 엔티티, 패턴 등을 표준화된 방법으로 표현할 때도 UML 프로파일을 참조하게 된다. 결국 UML 프로파일은 MDA의 PIM, PSM 수준에서 각 단계 별 설계 모델을 표현하기 위해서 필요한 추가적인 명세의 표준장치들로 구성되며 각 도구와 플랫폼은 이를 참조하는 역할을 하게 된다.

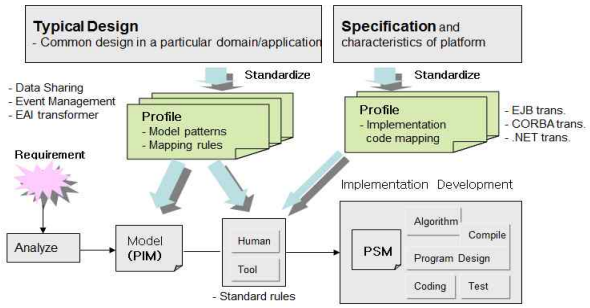

Fig. 1. Development approach to the model using the UML profile

1.2 EDOC을 위한 UML 프로파일

OMG에서는 컴포넌트 아키텍처에서의 이벤트, 프로세싱, 엔티티, 패턴 등을 표현하기 위해 분산 컴퓨팅 환경에 대해서 Enterprise Distributed Object Computing (EDOC)을 위한 프로파일(UML Profile for EDOC)을 제안하였다. 하지만, 이는 객체 풀링, 동기화 서비스, 트랙잭션 서비스 등의 시간 제약성과 보안 서비스 등의 기능을 표현하기 어려우며, 이들을 명세 하기 위한 스펙을 지원하고 있지 않아 PSM으로의 변환에 어려움이 있다. EDOC을 위한 프로파일(UML Profile for EDOC)은 총 7개의 스펙으로 구성된다. 첫째, Enterprise Collaboration Architecture (ECA)[6]은 모델링 프레임워크에 의해 EDOC 시스템을 개발하기 위한 아키텍처이다. 둘째, Metamodel and UML Profile for Java and EJB는 Java와 EJB에 관한 메타 모델이다. 셋째, Flow Composition Model (FCM)은 어플리케이션간의 정보 흐름과 상호관계를 명세할 수 있다. 넷째, UML Profile for Patterns[7]는 UML 패키지 표기법을 이용해서 비즈니스 기능 객체 패턴(Business Function Object Patterns)를 표현 할 수 있다. 다섯째, UML Profile for ECA는 엔티티, 이벤트, 비즈니스에 관한 스펙을 제공하며 이 중 엔티티에 관한 요소 매핑은 표 1과 같다. 여섯째, UML Profile for Meta Object Facility[8]는 UML과 MOF간의 매핑을 명세한다. 마지막으로 UML Profile for Relationships[9]은 비즈니스 모델에서의 관계에 관한 표준을 명세한다.

Table 1. Mapping in the UML Profile for ECA element

Meta Model Element	UML Profile Element	UML Base Class
Data Manager	Data Manager	Class
Entity Data	Entity Data	Class
Entity	Entity	Class
Entity Role	Entity Role	Class
Class Key	Key	Class
Key Element	Key Element	Attribute
Key Attribute	Key Attribute	Attribute
Foreign Key	Foreign Key	Attribute
Data Manager	Data Manager	Class

1.3 Wang의 Model to Model Transformation[10]

Wang은 모델 기반 시스템 엔지니어링을 지원하는 모델 대 모델 매핑 및 변환에 대한 자동화 방안을 제안한다. 이를 위해 의미와 구문 두 가지 측면에 대한 모델 변환 규칙을 정의하고, 모델 사이의 매핑 및 변환을 메타 모델 기반 변환 프로세스로 설명하고 있다. 또한, 제안된 도구는 정의된 규칙을 적용한 확장 모델 설계 기능을 사용하여 모델 생성과 검사를 자동화 할 수 있다. 확장 모델의 설계는 컴포넌트의 서비스 처리를 위한 Provided Ports와 다른 컴포넌트와 함께 협의하기 위한 Required Ports를 정의한다. 컴포넌트의 내부 설계는 클래스와 클래스들간의 관계를 묘사하고, 컴포넌트 논리적 배치는 분산된 아키텍처의 명세를 보인다. 하지만 제안된 기법과 확장 모델 생성을 위한 메타 모델이 OMG 표준인 MOF 등을 준수하지 않아 이식성과 상호 운영성이 부족하고, PSM의 구체적인 명세 요소와 지원 플랫폼은 언급하지 않았으며, 실시간 분산 아키텍처를 지원하기 위한 시간 제약성 등의 기능들에 대한 부분은 언급하지 않았다.

1.4 Oldevik의 Model to Text Transformations[11]

Oldevik은 OMG의 MOF Model to Text Transformation RFP와 MOF Query, View and Transformations (QVT) [12] 표준을 기본 기준으로 삼아 MOFScript 언어를 제안하고 이를 위한 이클립스 플러그-인 툴을 제안한다. 제안된 MOFScript 언어는 QVT, Object Constraint Language (OCL). MOF등의 표준 과 요구사항을 최대한 재 사용하고 충족시키며, 이를 증명하기 위하여 여러 가지 요구 특성에 대해 평가하였다. MOFScript 툴은 툴 컴포넌트와 서비스 컴포넌트의 두 가지 주요한 아키텍처로 구성되어 있는데, 툴 컴포넌트는 사용자가 모델을 편집할 수 있는 기능을 제공하고, 서비스 컴포넌트는 소스 코드 변환과 파싱 체크에 대한 기능을 제공한다. 제안된 MOFScript 언어와 툴은 OMG 표준을 최대한 준수하며, 사용 편리성과 확장성을 만족시키며 모델을 생성한다. 하지만 모델 생성을 위한 구체적인 UML 프로파일이나 OCL 등을 적용한 UML 확장 장치와 모델링에 대한 언급 없이 규칙과 템플릿만 제시하여 실제 사용하기에는 부족함이 있다. 또한 실시간 분산 아키텍처를 위한 동기화, 트랙잭션, 보안 서비스 등에 대한 모델 생성은 일부만 언급하였다.

1.5 닷넷 애트리뷰트 코드

닷넷 플랫폼에서는 일부 특성 형식을 미리 정의하여 런타임 동작을 제어하는 데 사용할 수 있다. 특성 형식을 미리 정의하여 닷넷 플랫폼의 Base Class Library (BCL)에서 직접적으로 표현되지 않는 언어 기능을 표현할 수 있으며, 사용자 정의 추가 특성 형식을 정의하고 사용할 수 있는데, 이것이 “애트리뷰트” 코드이다. 애트리뷰트 코드는 소스 코드안에서 타입, 메소드, 프로퍼티 등에 관련된 정보를 명시적으로 선언해 줌으로써 관련된 프로그램 엔터티가 런타임에 해당 특성을 갖도록 하는 강력한 방법을 제공한다. 애트리뷰트는 두 가지 종류로 나눌 수 있는데 하나는 Common Language Runtime's(CLR)의 BCL 에 정의되어 있는 기능을 수행하는 것과, 다른 하나는 사용자가 추가적인 정보를 코드상에 나타내기 위해 정의할 수 있는 사용자 정의 추가 특성 형식이 그것이다 [13].

그림 2는 최소의 닷넷 클래스 코드에서 트랜잭션과 객체 풀링, COM 등에 대한 속성을 애트리뷰트 코드를 이용해 지정한 예이다. 애트리뷰트는 이외에도 보안, 동기화, 적시 활성화 등의 속성을 지정 할 수 있다.

```
…
  [ComVisible(true)]
  [Transaction(TransactionOption.Supported)]
  [ObjectPooling(true, MinPoolSize=5,MaxPoolSize=10,
    CreationTimeout=5000)]

  public class MyComponent : ServicedComponent
  {
        public MyComponent()
        {
        …
        }
…
```

Fig. 2. NET Attribute code format

III. Service Element for Real-time Distributed architecture

본 장에서는 실시간 분산 아키텍처 기반의 엔터프라이즈 시스템 개발 시 필수적으로 고려할 기능 요소에 대한 메타 모델을 정의한다. 애플리케이션 아키텍처는 애플리케이션이 갖고 있는 요구사항에 따라 결정된다고 할 수 있다. 일반적인 최근 엔터프라이즈 애플리케이션에 대한 요구사항은 높은 성능, 편리한 사용자 화면은 기본에 높은 확장성과 안정성이 요구되며 이를 위해 컴포넌트 기반의 실시간 분산 아키텍처가 선택되어

지고 있다. 이러한 아키텍처는 객체의 생명 주기와 반응 시간에 따른 동기화, 트랜잭션, 객체 풀링, 보안 서비스 등의 기능을 필연적으로 사용하게 된다 [14].

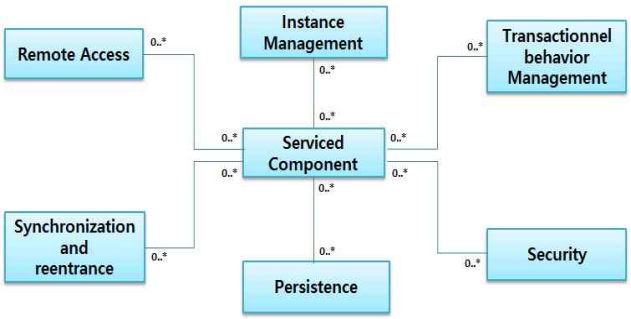

Fig. 3. Service Meta Model for Real-time distributed architecture

실시간 분산 아키텍처 기반 서비스를 지원하는 컴포넌트를 Serviced Component라고 할 때, 서비스 요소는 그림 3과 같이 크게 리모트 액세스, 인스턴스 관리, 트랜잭션 관리, 동기화, 지속성, 보안 서비스로 구성된다. 리모트 액세스는 지역적으로 떨어진 컴포넌트의 인스턴스를 호출하여 필요한 기능을 요청하는서비스이며, 인스턴스 관리란 인스턴스의 활성/비활성화, 풀링, 생명주기를 관리하는 서비스이다. 실시간 분산 아키텍처 기반 서비스를 지원하는 컴포넌트가 되고자 하는 클래스는 Serviced Component 클래스를 베이스 클래스로 삼으면 된다.

IV. RDA-.NET Profile

본 장에서는 앞에서 정의한 실시간 분산 아키텍처 기반 서비스 메타 모델을 기반으로 이들 서비스 명세를 지원하는 UML 프로파일과 .NET/C#용 UML 프로파일을 제안한다. 이 프로파일은 .NET/C# 관련 UML 프로파일을 가정하고, 여기에 본 논문에서 정의한 실시간 분산 아키텍처 기반 서비스 요소를 추가한 것으로 본 논문에서는 이를 “RDA-.NET profile(Real time Distributed Architecture Based .NET profile)”이라 정의한다.

4.1 실시간 분산 아키텍처 기반 서비스 명세를 위한 요소

표 2에서 트랜잭션 지원은 생성된 코드가 컴포넌트로서 트랜잭션 속성을 갖도록 명시적으로 표현한다. «Transaction» 스테레오 타입으로 명세한 클래스 혹은 컴포넌트는 트랜잭션 속성이 Supported가 되어 필요 시 트랜잭션에 참여하게 된다. 트랜잭션 종료는 «AutoComplete» 를 메소드에 명세함으로써 생성된 코드에서 메소드가 끝날 때 오류가 발생 했는지의 여부에 따라 해당 컴포넌트를 포함한 전체 트랜잭션을 커밋할 것인지 취소할 것인지를 결정하는 작업을 실행 시간에 자동으로 수행해 줄 수 있도록 하는 역할을 한다.

Table 2. UML profile element for specification of the services based Real-time Distributed architecture

Element Name	Specification Notation	UML Base Class	Remark
Transaction Support	«Transaction»	Class, Component	
Transaction Complete	«AutoComplete»	Method	
Security	{Security = “role name”, SetEveryoneAccess = value}	Class	OCL
Synchronization	«Synchronization»	Class	
Object Pooling	{Object Pooling = true, MinPoolSize=value, MaxPoolSize=value, CreationTimeout=value}	Class	OCL
Object Life Cycle	{Just In Time Activation}	Class	OCL
Queue use	«Queued»	Class, Component	

보안은 {Security = "role name", SetEveryoneAccess = value} 꼬리표 값을 사용하고, role name에는 보안에 관련된 정책 및 역할을 설정한 정책 명(role name)을 명세 한다. SetEveryoneAccess 속성은 모든 사용자 그룹에 대한 제어 권한에 대한 설정으로 속성을 true로 설정하면 자동으로 모든 사용자 그룹이 해당 역할에 추가된다.

객체 풀링은 필요한 컴포넌트의 인스턴스를 미리 만들어 풀(pool)에 넣어 놓고 클라이언트가 이 컴포넌트를 요구하면 새로운 컴포넌트를 만드는 것이 아니라 풀에 있는 객체를 활성화시키는 메커니즘을 말한다. 이에 대한 기능을 지원하는 코드를 생성하기 위하여 {Object Pooling = true, MinPoolSize = value, MaxPoolSize = value, CreationTimeout = value} 꼬리표 값을 사용한다. 최소 객체 수(MinPoolSize)는 COM+ 애플리케이션 시작과 함께 풀에 생성될 컴포넌트 인스턴스 개수를 말한다. 반면 최대 객체 수(MaxPoolSize)는 풀에 생성될 수 있는 최대 인스턴스 개수를 말한다.

적시 활성화(JIT: Just-In-Time Activation)란 컴포넌트가 작업을 완료하면 클라이언트가 컴포넌트에 대한 참조를 유지하고 있더라도 즉시 객체를 비활성화시키는 것을 말한다. 클라이언트가 이 컴포넌트에 대해 두 번째 메소드를 호출하면 새로운 컴포넌트의 인스턴스를 적시 활성화시킨다. 즉, 적시 활성화가 사용되면 클라이언트가 사용하는 컴포넌트는 메소드를 호출할 때마다 달라진다. 적시 활성화는 트랜잭션의 정확

함, 특히 일관성과 격리성을 지키는데 도움을 준다.

4.2 RDA-.NET 프로파일 중 .NET/ C# 명세를 위한 항목

EJB에 관한 UML 프로파일이 제정된 것에 반해 닷넷에 관한 UML 프로파일은 아직 정의되지 않은 상태이다. 이에 본 장에서는 닷넷용 PSM을 구성하는 필수 요소와 문법, 그리고 각 요소들의 구조를 정의한 메타모델을 제안한다. PSM 메타모델은 PSM 모델을 정의하는 메타모델로서 PSM 모델 변환 시 변환 대상이 되는 설계 모델요소를 정의한 메타모델이다.

PSM 메타모델 정의서는 PSM 모델을 구성하는 PSM 메타 모델 요소와 유형 및 각 요소 별 메타클래스를 정의하고 제약조건을 기술한다. PSM 메타모델은 표 3과 같이 계층별로 구분하여 PSM에서 표현하여야 하는 항목을 분류하며, “C# Operator”와 같이 모든 계층에서 공통으로 사용 가능한 요소는 Common 계층에 기술한다.

표 4는 RDA-.NET profile에서 .NET/C# 명세를 위한 UML 프로파일의 구성요소들을 정의한 것이다. 이들 프로파일 요소는 닷넷용 PSM 메타모델 요소를 확장하여 .NET/C# 용 PSM 모델 생성에 사용하기 위한 것이다.

4.3 RDA-.NET 프로파일 문서 내용

앞에서 정의한 RDA-.NET Profile의 메타모델과 구성 요소들을 실제 적용해 XMI/XML 기반의 확장된UML 프로파일 문서를 작성하였을 경우 프로파일의 일부 내용을 살펴보면 그림 4와 같다. 이 프로파일은 오픈소스 UML/MDA 도구인 StarUML의 플러그인으로 개발하여 실시간 분산 아키텍처 기반 서비스를 지원하는 닷넷용 PSM을 생성하는데 사용하게 된다.

Table 3. The Meta Model element of PSM for .NET

Layer	Meta Model Element	유형	Definition	Meta Class
Presentation	ASPX	Stereotype	Indicates User Interface	UML Class
Business	DotNet Assembly	Stereotype	A unit of reusable and deployable .NET components	UML Component
Persistent	Data Type	Stereotype	The dataset class in which the information is stored	UML Class
Common	C# Operator	Stereotype	Indicates C# Operator	UML Operation
…	…	…	…	…

Table 4. UML profile element for .NET/C# specification

Element Name	Specification Notation	Description	Base Classes
DotNet Assembly	«DotNetAssembly»	C# by compilation result of source file that . NET assembly	UML Component
CSharp SourceFile	«CSharpSourceFile»	C# source file that implementation code comes	UML Component
CSharp Delegate	«CSharpDelegate»	C# Delegate	UML Class
CSharp Struct	«CSharpStruct»	C# Struct type	UML Class
CSharp Event	«CSharpEvent»	When define C# event object	UML Operation
CSharp Property	«CSharpProperty»	Display that is C# Property that express attribute of class or structure	UML Operation
CSharp Indexer	«CSharpIndexer»	Display that is Indexer that can approach class or structure with general arrangement in C#	UML Operation
ASPX	«ASPX»	Display that is client side web page by .NET web page extension life	UML Class
Data Type	«DataType»	By objective that information is stored, is corresponded in mapping table to database	UML Class
…	…	…	…

```
<?xml version="1.0" encoding="EUC-KR" ?>
<PROFILE version="1.0">
  <HEADER>
    <NAME>RDA-.NET</NAME>
    <DISPLAYNAME>RDA-.NET Profile</DISPLAYNAME>
  </HEADER>
...
    <STEREOTYPELIST>
      <STEREOTYPE>
        <NAME>CSharpSourceFile</NAME>
        <DESCRIPTION> Source file with C# code
</DESCRIPTION>
        <BASECLASSES>
          <BASECLASS>UMLComponent</BASECLASS>
        </BASECLASSES>
      </STEREOTYPE>
...
    <STEREOTYPE>
      <NAME>Transaction</NAME>
     <DESCRIPTION> That Transaction is necessary class
     indication </DESCRIPTION>
        <BASECLASSES>
          <BASECLASS>UMLClass</BASECLASS>
          <BASECLASS>UMLComponent</BASECLASS>
        </BASECLASSES>
    </STEREOTYPE>

    <STEREOTYPE>
      <NAME>Synchronization</NAME>
     <DESCRIPTION> That Synchronization is necessary class
     indication </DESCRIPTION>
        <BASECLASSES>
          <BASECLASS>UMLClass</BASECLASS>
          </BASECLASSES>
    </STEREOTYPE>
...
<TAGDEFINITIONSETLIST>
  <TAGDEFINITIONSET>
     <NAME>RDA Service</NAME>
       <BASECLASSES>
         <BASECLASS>UML Class</BASECLASS>
       </BASECLASSES>
  <TAGDEFINITIONLIST>
    <TAGDEFINITION lock="False">
       <NAME>Security </NAME>
         <TAGTYPE>String</TAGTYPE>
    </TAGDEFINITION>

    <TAGDEFINITION lock="False">
       <NAME>ObjectPooling</NAME>
         <TAGTYPE>String</TAGTYPE>
    </TAGDEFINITION>
...
  </TAGDEFINITIONSET>
</TAGDEFINITIONSETLIST>
```

Fig. 4. The RDA-.NET Profile contents

4.4 RDA-.NET 프로파일을 적용한 컴포넌트 개발 프로세스

본 논문의 기법을 보다 실용적이고 사용하기 용이하도록 하기 위해 StarUML 기반의 프로토타입 툴을 개발한다. StarUML은 UML메타모델과 애플리케이션 객체 등 프로그램의 대부분에 접근할 수 있도록 모듈을 객체화하고 API를 외부로 노출시켜 쉽게 애드인을 개발하여 연동 시킬 수 있다. 또한 XML/XMI 기반의 프로파일을 프로젝트에 포함시키는 것만으로 소스 코드 수정 없이 외부 API를 통해서 모델링 요소의 태그값을 설정하거나 변경하는 것이 가능하다.

본 장에서는 기존의 MDA개발 프로세스에 앞에서 정의하고 작성한 RDA-.NET Profile과 StarUML, 그리고 StarUML 기반의 애드인을 적용한 컴포넌트 개발 프로세스를 그림 5와 같이 제안한다. 요구사항 분석 단계에서는 요구사항 명세서를 분석하여 기능적 요구사항과 비기능적 요구사항을 도출한다.

먼저 앞에서 정의한 대로 RDA-.NET Profile을 XML/XMI 형식으로 작성한 후 이것을 StarUML에서 사용할 수 있도록 애드인으로 개발한다. 이후 PIM 설계 단계에서는 실시간 분산 아키텍처 기반 서비스 요소를 애드인을 이용하여 설계한다. 본 논문에서는 RDA-.NET Profile을 이용한 플랫폼 독립적인 설계를 "RDA-PIM" 이라 정의한다. PSM 설계 단계에서는 개략 설계물인 RDA-PIM으로 프로파일에서 제공하는 명세 요소를 이용하여 닷넷 플랫폼을 고려한 상세 설계물인 닷넷용 PSM을 생성한다.

마지막으로 코드 생성 단계에서는 생성된 PSM을 RDA-.NET Profile과 개발한 애드인에 의해 닷넷 애트리뷰트 코드를 포함한 닷넷 컴포넌트 산출물을 생성한다. 본 논문에서는 코드 생성을 제외한 PIM, PSM 설계 단계까지를 연구 범위로 한다.

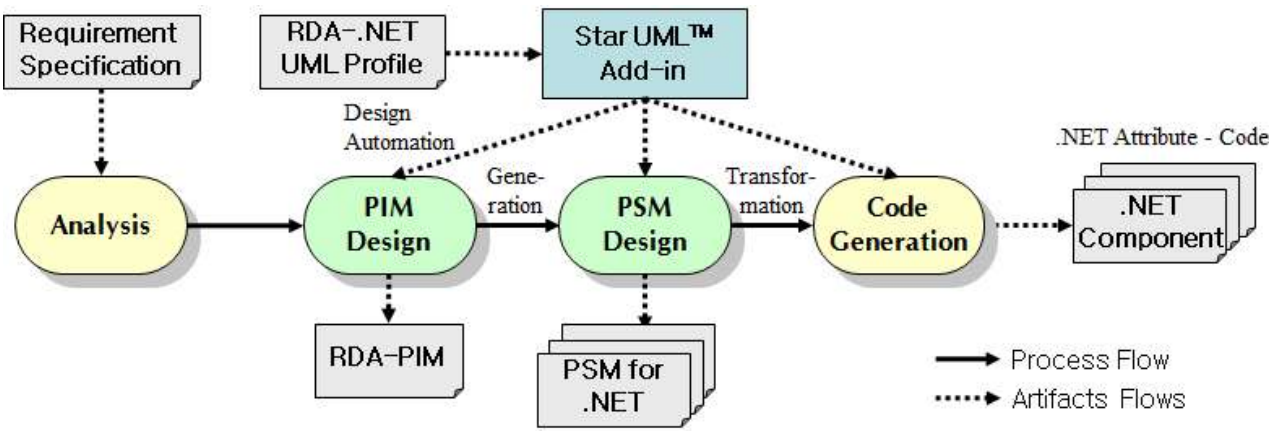

Fig. 5. Component development process applying the RDA-.NET profile

V. Case Study

본 논문에서 적용 사례는 자동차 임대 업무 사업을 하고 있는 BCC(Best Car Corporation)사의 영업 관리 시스템 중 대여/ 예약관리 기능에 대하여 적용하였다. 이는 StarUML, .NET/C#, XML/XMI를 기반으로 구현 하였다.

5.1 RDA-PIM 설계

먼저 앞에서 정의한 RDA-.NET Profile에 해당하는 아이콘 파일, 레지스트리 등록 파일 작성 등 필요한 작업을 한 후 그림 6과 같이 StarUML의 프로파일 매니저를 통해 새로운 프로젝트에 프로파일을 포함 시킨다.

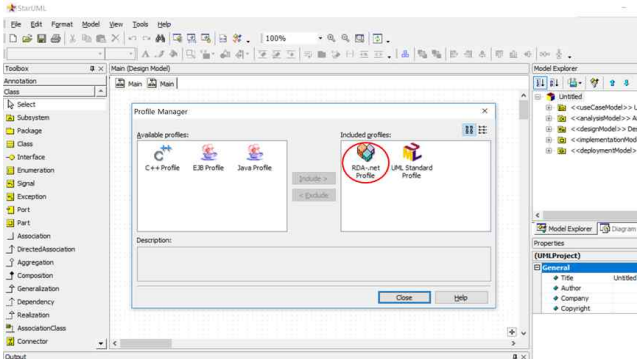

Fig. 6. Image that RDA-.NET profile included in the new project

RDA-.NET profile을 포함시킨 후에는 개발하려는 시스템에 대한 PIM을 설계한다. 이때 앞에서 정의한 실시간 분산 아키텍처 기반 서비스 명세 요소를 개발한 애드인을 이용하여 필요한 클래스나 컴포넌트 혹은 메소드에 생성한다.

그림 7은 RDA-.NET profile의 스테레오 타입과 태그정의 항목의 내용을 애드인의 확장 속성 편집기를 이용하여 속성값을 설정해 주는 것을 나타낸다. 설정한 값은 해당 모델 요소에 정의된 기준에 따라 스테레오 타입, 꼬리표 값, OCL 등의 형태로 자동 생성된다.

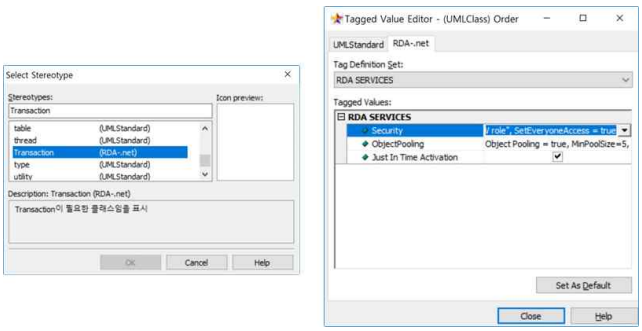

Fig. 7. Set the value of elements that defined in the RDA-.NET profile

그림 8은 'Rental'클래스가 트랜잭션 속성을 지원하며 'checkOut' 메소드는 트랜잭션 종료에 대한 속성이 'AutoComplete' 속성을 지원하도록 생성되었으며, 객체의 활성/ 비 활성화 속성이 'Just In Time Activation' 으로 생성 된 것을 보여준다. 'CarItem' 클래스에는 객체 풀링에 대한 속성값이 OCL 형태로 생성되었다.

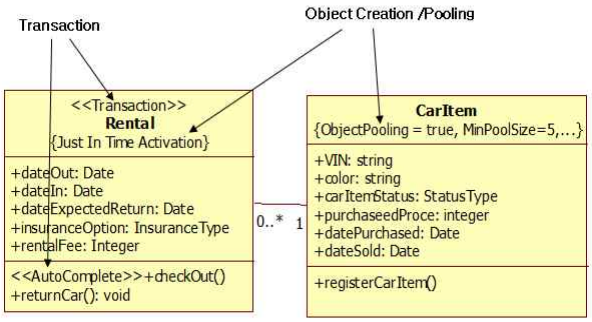

Fig. 8. Example of generate RDA-PIM

5.2 닷넷용 PSM 생성

RDA-PIM을 생성한 후에는 RDA-.NET Profile의 .NET/C# 명세를 위한 항목들을 이용하여 닷넷용 PSM을 생성하게 된다. 이를 위하여 PIM에서 PSM로의 변환을 위해 필요한 변환 규칙들과 제약조건을 정의한다. 각각의 변환규칙은 실시간 분산 아키텍처 기반 서비스와 PSM 메타모델 정의서에 작성된 각 요소간의 변환 관계를 정의함으로써 작성된다. 표 5와 같이 변환규칙 정의서는 각 메타모델 구성요소와 변환유형, 변환규칙, 제약조건으로 구성된다.

Table 5. The example of transformation rule definition

Meta Model Element for PSM	Transfor-mation category	Transformation rule	Constraints
PSM configuration elements	Class, Operation, Create attributes, etc.	The description of transformation rule applied during model transformation	The description of constraint for transformation rules
…	…	…	…

OMG에서 제정한 MOF를 기반으로 하는 모델 간의 변환작업을 위한 QVT는 모델 생성을 위한 언어, 모델 쿼링을 위한 언어, 변환규칙을 기술하기 위한 변환 정의 언어로 구성된다. QVT는 아키텍처, QVT common structure, QVT using a rule-based language, QVT in the context of UML Profiles, QVT using a framework approach, QVT interoperability, 추적성의 전체 7가지 영역으로 나뉘어 정의된다 [15]. 또한, 타겟모델이 생성되지 않은 경우에 생성 시 필요한 조건과 이미 생성된 경우에 변환규칙을 적용하기 위해 소스모델이 포함해야만 하는 조건을 기술한 것을 소스/타켓 언어 조건(source/target language condition)이라고 한다 [16].

그림 9는 QVT를 기반으로 PIM모델을 닷넷용 PSM으로 매핑할 때 사용되는 정의인데 RDA-.NET Profile에 정의된

RDAService 요소는 닷넷 애트리뷰트 클래스로, .NET/C# 요소는 표3의 닷넷용 PSM 메타모델 정의서 항목으로 변환시키겠다는 정의를 하고 있다. 소스/타겟 조건은 OCL Boolean 표현 식으로 작성한다 [16].

```
Transformation RDAServiceToAttributeClass (UMLProfile,
.NET) {
      source umlProfileElement    : UMLProfile ::
RDAService;
       target attributeClass          : .NET :: AttributeClass;
       unidirectional;
       mapping
          umlProfileElement.name     <~>
.NETAttributeClass.name
          umlProfileElement.parameter     <~>
.NETAttributeClass.parameter
    }
...

  S := GetStringTaggedValue(AClass, 'RDA-.NET', 'RDA
SERVICES', 'transaction');
     if S <> '' then begin
       FWriter.WriteLine('[Transaction(%s)]', [S]);
       PropAdded := True;
     end
     else if AClass.StereotypeName = 'Transaction' then begin

FWriter.WriteLine('[Transaction(TransactionOption.Supported)]');
       PropAdded := True;
     end;

     S := GetStringTaggedValue(AClass, 'RDA-.NET', 'RDA
SERVICES', 'ObjectPooling');
     if S <> '' then begin
       FWriter.WriteLine('[ObjectPooling(%s)]', [S]);
       PropAdded := True;
     end
     else if AClass.StereotypeName = 'Object Pooling' then
begin
       FWriter.WriteLine('[ObjectPooling(true,
MinPoolSize=x,MaxPoolSize=y, CreationTimeout=z)]');
       PropAdded := True;
     end;
...
```

Fig. 9. The realization of transformation rule using transformation definition language

이렇게 명세한 내용들은 궁극적으로 코드 자동 생성을 통하여 실시간 분산 아키텍처 기반 서비스를 지원하는 COM+ 컴포넌트 용 닷넷 클래스가 될 수 있다.

RDA-PIM을 모델링 한 후 변환 규칙을 구현한 애드인을 통하여 닷넷용 PSM을 생성한 결과는 그림 10과 같다. 그림 10의 예제는 BCC사 영업 관리 시스템 중 대여/ 예약관리 모듈에 대한 클래스 다이어그램으로 RDA-.NET Profile에 정의된 RDAService 요소와 닷넷용 PSM 메타모델 정의서 항목들이 생성된 것을 볼 수 있다.

VI. Assessment

기존 연구와 비교해 보면, Wang의 Model to Model Transformation[10]은 의미와 구문 두 가지 측면에 대한 모델 변환 규칙을 정의하고, 모델 사이의 매핑 및 변환을 메타모델 기반 변환 프로세스로 설명하고 있으며, 정의된 규칙을 적용한 확장 모델 설계가 가능함을 보였다. 하지만 제안된 기법과 확장 모델 생성을 위한 메타 모델이 OMG 표준인 MOF 등을 준수하지 않아 이식성과 상호 운영성이 부족하고, 엔터프라이즈 시스템 개발에 필수적인 기능인 실시간 분산 아키텍처 기반 서비스에 대한 모델 변환 역시 언급하지 않았다. Oldevik[11]은 OMG의 QVT 표준을 기본 기준으로 삼아 MOFScript 언어와 이를 위한 이클립스 플러그-인 툴을 제안한다. 제안된 MOFScript 언어와 툴은 OMG 표준을 최대한 준수하고, 사용 편리성과 확장성을 만족시키며 모델을 생성한다. 하지만 구체적인 프로파일이나 OCL등을 적용한 UML 확장

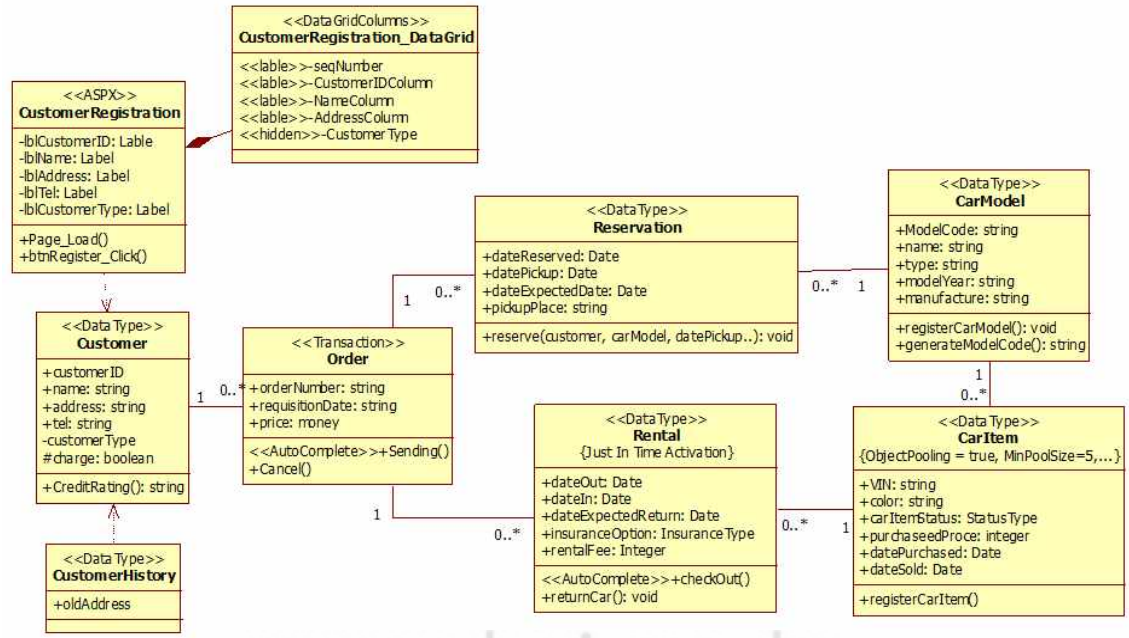

Fig. 10. Rental/Reservation administration class diagram(PSM for .NET)

장치와 모델링에 대한 언급이 없으며, 실시간 분산 아키텍처 기반 서비스에 대한 모델 생성은 일부만 언급하였다.

본 논문에서는 실시간 분산 아키텍처 기반 서비스와 .NET/C# 명세를 위한 요소들을 제안하고, 이 요소들을 정의한 메타모델을 준수하는 UML 프로파일로 정의하여 RDA-PIM과 닷넷용 PSM과 같은 확장 모델을 생성할 수 있는 장치를 만들었다. 또한, 이를 오픈소스 UML/MDA 도구의 애드인으로 개발 함으로써 실시간 분산 아키텍처 기반 서비스를 지원하는 닷넷 컴포넌트의 자동 생산이 가능하게 하였다.

본 논문에서 제안한 기법을 평가하기 위하여 다음과 같은 평가항목을 선정하였다. 평가 항목의 선정 기준은 OMG의 MDA 가이드 1.0 [17] 준수 여부, OMG의 MOF, XMI, QVT 등 표준의 준수 및 사용 여부와 닷넷 플랫폼 및 실시간 분산 아키텍처 기반 서비스에 대한 고려, 기법의 실용성 및 효과성, 효율성 등에 근거하고 있다.

- OMG 표준인 MDA 가이드 1.0 준수 여부: MDA 가이드 1.0의 지침 및 원리를 준수하는지에 대한 평가
- OMG 표준인 MOF, XMI, QVT 준수 및 사용 여부: 제안한 기법이 모델 변환을 위한 OMG 표준인 MOF, XMI, QVT를 사용하며, 요건을 준수하는지에 대한 평가
- UML Profile의 제공 여부: PIM 모델 명세 및 PSM으로의 변환을 위한 XML/XMI 기반의 UML Profile을 제공하는지에 대한 평가
- .NET/ C# 용 UML Profile의 지원 여부: .NET/ C# 용 PSM 모델 명세 및 확장 모델 생성을 위한 UML Profile을 지원하는지에 대한 평가
- 실시간 분산 아키텍처 기반 서비스 요소 명세 및 이를 지원하는 모델 생성 여부: 실시간 분산 아키텍처 기반 서비스를 명세하기 위한 장치 및 이를 지원하는 모델 생성 여부에 대한 평가
- 도구의 제공 여부: 모델 명세 및 변환을 위한 도구를 제공하는지에 대한 평가
- 기법에 대한 상세지침 제공 여부: 기법을 수행하기 위한 세부적인 기준과 지침, 프로세스가 제공되는지를 평가
- 적용 예제 제공 여부: 활동의 지침을 적용한 사례를 제공함으로써 제안된 프로세스의 이해를 높여 적용 성을 향상시킬 수 있는지를 평가

이상의 평가 항목으로 표 6에서 OMG의 EDOC 프로파일과 Kath의 Executable Models, Oldevik의 Model to Text Transformations 모델을 대상으로 본 기법을 비교해 보았다.

Table 6. Comparison with related research

Technology / Criterion	EDOC Profile	Wang	Oldevik	Proposed Technique
Compliant with OMG Standard MDA Guide	○	○	○	○
Compliant with OMG standards MOF, XMI, QVT and use	○		○	○
Providing UML Profile	○			○
Support UML Profile for .NET/C#				○
Specification of service elements based on Real-time Distributed architecture and generation of models supporting them	△	△	△	○
Providing tools		○		○
Providing detailed guidelines for proposed technique				○
Providing application example		○		○

VII. Conclusions

본 논문은 MDA기반 개발 방식에서 실용적이고 재사용 가능한 확장 모델을 생성하는 방안에 대한 연구로써 현재까지 발표된 UML 프로파일의 경우 PIM에서 PSM을 생성 하거나 PSM에서 소스코드를 생성하였을 시 실시간 분산 아키텍처 기반 서비스에 대한 명세 및 모델 생성을 위한 장치가 없다는 점과 닷넷용 UML 프로파일이 아직 제정되지 않았다는 점에서 출발하였다. 현재까지의 MDA 연구는 다양한 변환 포맷 지원이나 기본 베이스 모델의 소스 변환은 성공적으로 이루어 지고 있으나 생성된 소스 코드는 엔터프라이즈 애플리케이션에서 필요한 실시간 분산 아키텍처 기반 서비스는 지원하지 않고 있으며, 비 효율적인 면이 여럿 남아있다.

본 논문에서는 엔터프라이즈 애플리케이션에서 필수적인 트랜잭션, 보안, 동기화 서비스, 객체 풀링 등의 실시간 분산 아키텍처 기반 서비스와 닷넷 플랫폼에 대한 확장 모델 생성을 위하여 RDA-.NET profile을 제안 하였다. 또한 이를 오픈소스 모델링 플랫폼인 StarUML에 적용할 수 있도록 구성하고, 정의된 메타 모델과 변환 규칙 등을 적용한 애드인을 개발하여 RDA-PIM과 닷넷용 PSM을 생성하여 보았다. RDA-.NET profile은 XMI 기반의 XML문서로서 쉽게 내용을 추가하거나 수정할 수 있으며, 개발한 애드인은 외부 API를

이용하여 쉽게 이를 적용할 수 있다. 또한 RDA-.NET profile은 OMG의 UML Profile의 기능을 그대로 지원하며, MOF를 준수하므로 MOF를 준수한 UML 도구 및 MDA 도구에서 사용이 가능하다. 이와 같은 장점들로 인해 본 논문에서 제안한 방법을 사용할 경우 닷넷 플랫폼과 실시간 분산 아키텍처에서 고려할 기능에 대한 하위 레벨의 정보를 자세히 알지 못하더라도 쉽고, 빠르게 재사용 가능한 확장 모델을 생성할 수 있어 설계 모델의 생산성, 확장성, 이식성 및 유지보수성을 증가 시킬 수 있다.

본 연구는 향후에 개발한 애드인을 통해 설계 모델의 확장 모델 생성 뿐 아니라 소스 코드 자동 생성에 대한 연구가 추가적으로 필요하다. 이를 위해 제안된 프로파일과 메타 모델을 보다 구체적으로 확장하고, 변환 규칙을 최적화하는 연구를 진행 중이다.

REFERENCES

[1] Nam-Yong Lee, and C.R. Litechy, “An empirical study of software reuse with Special Attention to Ada”, Software Engineering, IEEE Transactions , 1997

[2] YueHua. Lin,Jeff Gray "A Model Transformation Approach to Automated Model Transformation", Ph.D Thesis, 2007.

[3] OMG. Metamodel and UML Profile for Java and EJB Specification. February 2004. Version 1.0, formal/04-02-02. An Adopted Specification of the Object Management Group, Inc.

[4] https://sourceforge.net/projects/staruml/.

[5] Frankel, D., Model Driven Architecture™:Applying MDA™ to Enterprise Computing, Wiley, 2003.

[6] OMG, UML Profile for Enterprise Collaboration Architecture (ECA) V1.0, 2004.

[7] OMG, UML Profile for Patterns V1.0, 2004.

[8] MOF Model to Text Transformation Language RFP, OMG document ad/04-04-07.

[9] OMG, UML Profile for Relationships V1.0, 2004.

[10] Wang, T., Truptil, S. and Benaben, F., "An automatic model-to-model mapping and transformation methodology to serve model-based systems engineering", Information Systems and e-Business Management, Vol. 14, No. 1, pp. 1-14, June 2016.

[11] Jon Oldevik, Tor Neple, Roy Grønmo, Jan Aagedal, and Arne-J. Berre, "Toward Standardised Model to Text Transformations", Proceedings of the first European Conference (ECMDA'05), Springer Verlag Lecture Notes in Computer Science Vol. 3748 (LNCS 3748), November 2005.

[12] MOF 2.0 Query / Views / Transformations RFP, OMG document ad/2002-04-10.

[13] Lowy, Juval. COM and .NET Component Services. O'Reilly, 2001. 384 p.

[14] David S. Platt, Understanding COM+, Microsoft Press, 1999.

[15] MOF 2.0 Query / Views / Transformations RFP, OMG document ad/2002-04-10.

[16] Kleppe, A., Warmer, J. and Bast, W., MDA Explained, Addison-Wesley, 2003.

[17] OMG, MDA Guide Version 1.0.1, omg/2003-06-01, June 2003.

Authors

Deuk Kyu Kum received the M.S. and Ph.D. degrees in Computer Science and Engineering from Soongsil University, Korea, in 2005, 2012, respectively.

Dr. Kum joined the faculty of the Department of Computer Software at Dong Seoul University, Seongnam-si, Gyeonggi-do, Korea, in 2009. He is currently a Adj. Professor in the Department of Computer Software, Dong Seoul University. He is interested in big data analysis technology, internet and mobile computing, and cloud computing.