MDA/PSM상에서 퍼베이시브 서비스를 지원하는 닷넷 컴포넌트의 명세 및 생성 기법

(A Technique to Specify and Generate .NET Components in MDA/PSM for Pervasive Service)

금 득 규 † 김 수 동 ††

(Deuk Kyu Kum) (Soo Dong Kim)

요 약 컴포넌트 기술은 재사용 가능한 컴포넌트를 조합하여 효율적으로 소프트웨어 시스템을 개발하기 위한 기술로 정착되어 왔으며, 마이크로소프트의 닷넷은 최근의 대표적 컴포넌트 기술 중의 하나이다. 모델기반 아키텍처(Model Driven Architecture, MDA)는 설계 모델을 점진적으로 변환하여 소프트웨어를 자동으로 생성하는 새로운 개발 방식이다. MDA에서 구조적 모델 변환은 성공적으로 적용되었으나, 동적 모델과 퍼베이시브 서비스(Pervasive Services) 특히, 트랜잭션 서비스, 보안 서비스, 동기화 서비스, 객체 풀링 등과 같은 기능에 대한 모델 변환은 부족하다. 엔터프라이즈 애플리케이션 시스템은 다 계층 분산 아키텍처를 가지며 이러한 아키텍처에서 퍼베이시브 서비스는 필수적이다. 닷넷 플랫폼은 코드상에서 애트리뷰트(Attribute) 코드를 명시함으로써 이러한 퍼베이시브 서비스를 지원하는 Component Object Model+ (COM+) 컴포넌트를 구현할 수 있다. 본 논문에서는 엔터프라이즈 시스템 개발에서 필수적인 퍼베이시브 서비스의 기능과 닷넷 컴포넌트 생성을 위한 요소를 명세화하여, 이를 UML 프로파일로 정의한다. 또한, 정의된 프로파일을 이용하여 .NET/C#용 플랫폼 종속적 모델(PSM)을 명세한 후 도구를 이용하여 코드를 자동 생성하는 기법을 제안한다. 본 논문에서 정의된 UML 프로파일은 Meta Object Facility (MOF)를 준수한 UML 도구 및 MDA 도구에서 사용이 가능하다. 또한, 제안한 방법을 사용할 경우 퍼베이시브 서비스 기능을 지원하는 .NET 컴포넌트를 쉽게 자동 생성할 수 있으며 높은 개발 생산성, 확장성, 이식성 및 유지보수성을 증가시킬 수 있다.

키워드 : 모델 기반 아키텍처, UML 프로파일, 닷넷, COM+, 애트리뷰트 코드

Abstract Component technology has been widely accepted as an effective way for building software systems with reusable components, and Microsoft (MS) .NET is one of the recent representative component technologies. Model Driven Architecture (MDA) is a new development paradigm which generates software by transforming design models automatically and incrementally. Transformation of structural models in MDA has been successfully applied. However, transformation of dynamic models and pervasive services, such as transaction service, security service, synchronization service and object pooling are largely remains as an area for further research. The recent enterprise system has multi tier distributed architecture, and the functionality of early mentioned pervasive services is essential for this architecture. .NET platform can implement Component Object Model+ (COM+) component for supporting pervasive services by specify Attribute code. In this paper, we specify the functionalities of the COM+ component offering pervasive services, and then those functionalities are defined by UML profile. By using the profile, the Platform Specific Model (PSM) for .NET/C# is specified, and .NET components are automatically generated through our tool. The development productivity, extensibility, portability, and maintenance of software can be dramatically improved by using of the proposed methods.

Key words : Model Driven Architecture (MDA), UML Profile, Microsoft .NET, COM+, Attribute code

---

· 본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다. (UD060048AD)

† 학생회원 : 숭실대학교 컴퓨터학과
dkkum@otlab.ssu.ac.kr

†† 종신회원 : 숭실대학교 컴퓨터학과 교수
sdkim@ssu.ac.kr

논문접수 : 2006년 2월 6일
심사완료 : 2007년 6월 5일

1. 서 론

컴포넌트 기술은 재사용 가능한 컴포넌트를 조합하여 효율적으로 소프트웨어를 개발함으로써 개발노력과 시간을 줄여주는 기술로 널리 받아들여지고 있으며, 마이크로소프트의 닷넷은 최근의 대표적 컴포넌트 기술중의 하나이다. 모델 변환을 통하여 소프트웨어 개발을 자동화하는 기술인 MDA는 OMG가 지금까지 추진한 여러 표준화 작업을 바탕으로 한 모델 중심의 시스템 명세와 개발에 대한 것으로 설계 모델을 점진적으로 변환하여 소프트웨어를 자동으로 생성하는 새로운 개발 방식이다[1].

현재 Meta Object Facility (MOF) 기반의 모델을 소스 코드로 변환하는 여러 연구가 진행되고 있다. 하지만 아직까지 모델 변환을 위한 명확한 표준이나 벤더간 호환성이 부족하여 성능이나 편의성, 플랫폼 통합에는 약하다. 또한 구조적 모델에 비해 동적 모델과 엔터프라이즈 애플리케이션 개발에 필수적인 트랜잭션 서비스, 보안 서비스, 동기화 서비스, 객체 풀링 등의 퍼베이시브 서비스에 대한 모델 변환은 많이 부족하다. 최근의 엔터프라이즈 애플리케이션 아키텍처는 다 계층 분산 애플리케이션 아키텍처를 가지며, 이러한 아키텍처에서는 앞에서 언급한 퍼베이시브 서비스의 기능이 필수적이다[2]. 대개 이러한 기능을 처리하기 위해 미들웨어 제품을 사용하는데, 닷넷 환경에서는 COM+를 사용한다[3]. 마이크로소프트의 닷넷 플랫폼에서는 애트리뷰트 코드를 명시함으로써 이러한 퍼베이시브 서비스를 지원하는 COM+ 컴포넌트를 구현할 수 있다. Enterprise Java Beans(EJB) 기술을 위한 메타모델과 UML 프로파일이 발표[4]된 것에 비해 닷넷은 아직 제정되어 있지 않다.

본 논문에서는 퍼베이시브 서비스에 비견되는 COM+ 서비스의 메타모델을 제안하고, 퍼베이시브 서비스를 지원하는 닷넷 컴포넌트를 생성하기 위한 UML 프로파일을 정의한다. 또한, 정의된 프로파일을 이용하여 .NET/C#용 플랫폼 종속적 모델(PSM)을 명세한 후 오픈 소스 모델링 플랫폼인 "StarUML[5]"과 저자가 개발한 프로토타입 도구를 이용하여 코드를 자동 생성하는 기법을 제안한다. 제안한 프로파일은 OMG의 UML 프로파일 및 MOF 표준을 준수하므로, MOF를 준수한 UML 도구 및 MDA 도구에서 사용이 가능하며, 기존 연구에서 고려되지 않았던 퍼베이시브 서비스 요소를 쉽게 도구와 연동하여 PSM을 모델링할 수 있다. 또한, 제안한 방법을 사용할 경우 퍼베이시브 서비스 기능을 지원하는 .NET 컴포넌트를 쉽게 자동 생성할 수 있으며 높은 개발 생산성, 확장성, 이식성 및 유지보수성을 증가 시킬 수 있다.

2장에서는 UML 프로파일과 닷넷 애트리뷰트 코드를 소개하고 관련연구를 서술하며, 3장에서는 엔터프라이즈 시스템 개발을 위한 퍼베이시브 서비스와 이를 지원하기 위한 요소를 알아보고, 4장에서는 퍼베이시브 서비스를 적용한 .NET/C#용 UML 프로파일을 정의하고, 정의한 프로파일을 이용한 변환 과정 및 표현법을 설명한다. 5장에서는 사례연구를 기술하고, 6장은 본 논문을 평가하고, 마지막 7장은 결론을 내린다.

2. 관련 연구

2.1 UML 프로파일(Profile)

UML 프로파일은 설계 혹은 구현 모델에서 UML 기반의 설계를 MDA의 PIM이나 PSM 모델로 표현하기 위해 필요한 추가적인 명세의 표준장치들로 구성된다. UML 프로파일은 스테레오 타입(Stereo type), 제약 사항(Constraint), 꼬리표 값(Tagged Value) 및 아이콘으로 구성된다[6]. 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. 그리고 제약 사항과 데이타 타입을 추가적으로 정의할 수 있도록 허용한다. UML 프로파일은 플랫폼 종속적인 정보를 표현하기 위한 명세 방법과 코드 생성시 필요한 추가적인 상세 설계 정보의 명세 방법을 제공한다.

MDA는 PIM 수준에서 특정 도메인에 일반적인 요소들, 예를 들면 컴포넌트 아키텍처에서의 이벤트, 프로세싱, 엔티티, 패턴 등을 표준화된 방법으로 표현할 필요가 있다. 이러한 목적으로 분산 컴퓨팅 환경에 대해서 Enterprise Distributed Object Computing(EDOC)을 위한 프로파일(UML Profile for EDOC)을 OMG에서 제안하였다. 이 또한 도메인 영역이 늘어감에 따라 UML 프로파일이 늘어날 것이다.

현재 OMG에서는 PIM에 사용할 수 있는 UML 프로파일로 표 1과 같이 UML Profile for EDOC, UML Profile for Enterprise Application Integration(EAI) [7], UML Profile for Schedulability, performance and time 등을 이미 표준으로 제정했거나 진행 중이며 PSM에 사용할 수 있는 프로파일로는 UML Profile for CORBA[8], UML Profile for EJB 등이 OMG 혹은 기타 단체에서 제정했거나 진행 중이다.

2.2 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 표

표 1 현재 발표된 UML 프로파일

프로파일	정의
UML Profile for CORBA	CORBA를 PSM에서 UML로 표현하기 위한 Profile
UML Profile for EJB	EJB를 PSM에서 UML로 표현하기 위한 Profile
UML Profile for EDOC	Enterprise Application을 PIM 수준에서 기술하기 위하여 Process, Event, Pattern, Enterprise Collaboration, Architecture 등의 UML 표현 규칙을 정한 Profile로 7개의 스펙으로 구성된다.
UML Profile for EAI	Loosely-coupling System을 위한 PIM 수준의 Profile
UML Profile for Schedulability, Performance, and Time	Real-time System의 Schedulability, Performance 등을 모델 수준에서 정량적으로 분석 가능하도록 표현하기 위한 Profile

현할 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 BCL에 정의되어 있는 기능을 수행하는 것과, 다른 하나는 사용자가 추가적인 정보를 코드상에 나타내기 위해 정의할 수 있는 사용자 정의 추가 특성 형식이 그것이다[9].

애트리뷰트 클래스에서 모든 특성은 애트리뷰트에서 직접 또는 간접적으로 파생되며, 특성은 모든 대상 요소에 적용할 수 있다. 그림 1은 COM+ 컴포넌트가 되기 위한 최소의 닷넷 클래스 코드에서 트랜잭션과 객체 풀링, COM 등에 대한 속성을 애트리뷰트 코드를 이용해 지정한 예이다. 애트리뷰트는 이외에도 보안, 동기화, 적시 활성화 등의 속성을 지정할 수 있다.

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

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

그림 1 닷넷 애트리뷰트 코드 형식

2.3 Kath의 Executable Models[10]

Kath는 시스템의 구조적 모델이 아닌 동적 모델의 변환 방안을 제안한다. 이를 위해 두 가지 도구 연결고리를 개발했는데, 하나는 CORBA/CCM 플랫폼을 위한 것이고, 다른 하나는 웹 서비스 플랫폼을 위한 것으로 둘 모두 오픈 모델링 기반구조에 기초하고 있다. 이 오픈 모델링 기반구조에서는 MOF가 근간이 되는 기초 기술로 사용되며, 이 기반 구조를 사용하여 구현한 제품을 "medini"라 부른다. medini는 PIM 모델링 도구와 PSM 모델링 도구 그리고 코딩을 위한 통합 프로그래밍 환경이 있고, 이들은 이 아키텍처의 중앙에 위치한 레파지토리들로 연결되어 있으며, 레파지토리들은 모델 변환기를 통해 연결되어 있다. 이 도구들을 이용한 연구에서는 CCM과 웹 서비스를 PSM으로 명세 한 후 작성한 모델이 MOF기반의 메타모델로 변환되어 레파지토리에 저장 되어지고, 이를 모델 변환기에서 각각의 플랫폼에 따라 CCM의 IDL, 포트, 컴포넌트 등과 웹 서비스를 위한 코드를 생성하는 프로세스를 설명하고 있다. 하지만 제안된 프로세스가 추상적이고, 생성된 코드의 구체적인 형태나 방법은 언급하지 않았으며, 닷넷 플랫폼에 대한 지원 여부도 불확실하다. 또한 엔터프라이즈 시스템 개발에 필수적인 기능인 퍼베이시브 서비스에 대한 모델 변환은 언급하지 않았다.

2.4 Oldevik의 Model to Text Transformations[11]

Oldevik은 OMG의 MOF Model to Text Transformation RFP[12]와 MOF Query, View and Transformations(QVT)[13] 표준을 기본 기준으로 삼아 MOFScript 언어를 제안하고 이를 위한 이클립스 플러그-인 툴을 제안한다. 제안된 MOFScript 언어는 QVT, Object Constraint Language(OCL). MOF등의 표준과 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 모델 편집을 할 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 소스 코드를 생성한다. 하지만 구체적인 프로파일이나 OCL 등을 적용한 UML 확장 장치와 모델링에 대한 언급 없이 규칙과 템플릿만 제시하여 실제 사용하기에는 부족

함이 있다. 또한 퍼베이시브 서비스에 대한 모델 변환은 일부만 언급하였다.

3. 퍼베이시브 서비스 요소

본 장에서는 퍼베이시브 서비스를 지원하는 닷넷 컴포넌트를 생성하기 위한 UML 프로파일 요소를 정의하고 제안한다. 엔터프라이즈 애플리케이션 제작에 있어서는 보안, 동기화, 객체 풀링 등의 기능을 거의 필연적으로 사용하게 된다[2]. MDA에서는 엔터프라이즈 애플리케이션을 개발하는데 필요한 이러한 기능들을 '퍼베이시브 서비스'라 하고 있다.

3.1 COM+에서 제공하는 퍼베이시브 서비스

대개 퍼베이시브 서비스 기능을 지원하기 위해 미들웨어 제품을 사용하는데, 닷넷 환경에서는 COM+를 사용한다. COM+는 트랜잭션 처리나 컴포넌트들 사이의 로드 밸런싱, 보안 수준, 동기화 서비스, 객체 풀링 등을 관리해주는 일종의 플랫폼이다. COM+ 컴포넌트가 되고자 하는 클래스는 Serviced Component 클래스를 베이스 클래스로 삼으면 된다. 즉, Serviced Component 클래스에서 파생된 클래스는 모두 COM+ 컴포넌트가 될 자격을 갖는다. 그림 2는 Serviced Component에서 파생한 닷넷 컴포넌트가 제공하는 주요 COM+ 서비스를 요약한 것이다. 보다 상세한 COM+ 서비스와 닷넷 컴포넌트 기술에 대한 내용은 [14], [15]에서 찾아 볼 수 있다. 본 논문에서는 이후에 논의할 UML 프로파일과 관련된 메타모델을 제시함으로써 쉽게 이해할 수 있도록 하는데 목적을 두고 있다.

3.2 퍼베이시브 서비스 명세를 위한 요소

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

보안은 {Security = "role name", SetEveryoneAccess = 값} 꼬리표값을 사용하고, role name에는 보안에 관련된 정책 및 역할을 설정한 정책 명(role name)을 명세한다. 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 만

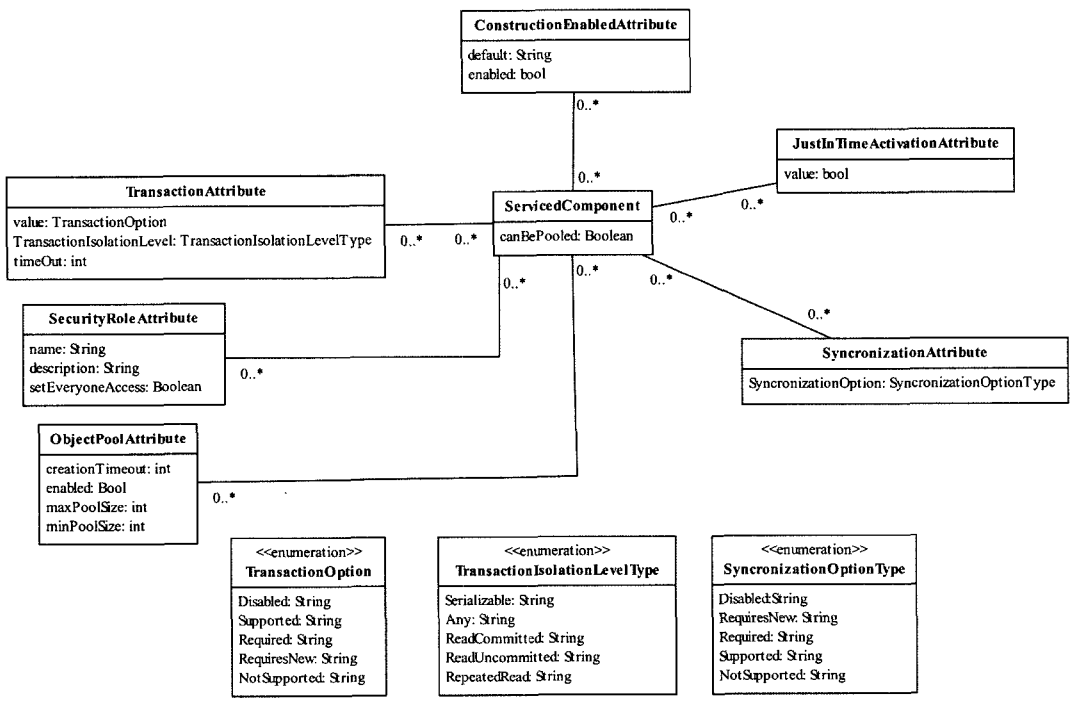

그림 2 닷넷 컴포넌트의 COM+ 서비스 메타모델

표 2 퍼베이시브 서비스 명세를 위한 UML 프로파일 요소

요 소	명세 표기법	적용되는 UML 요소	비고
트랜잭션 지원	«Transaction»	클래스, 컴포넌트	
트랜잭션 종료	«AutoComplete»	메소드	
보안	{Security = "role name", SetEveryoneAccess = 값}	클래스	OCL
동기화	«Synchronization»	클래스	
객체 풀링	{Object Pooling = true, MinPoolSize=값, MaxPoolSize=값, CreationTimeout=값}	클래스	OCL
객체생명주기 (적시활성화)	{Just In Time Activation}	클래스	OCL
큐 활용	«Queued»	클래스, 컴포넌트	
Com+컴포넌트	«Com+»	클래스, 컴포넌트	

들어 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=값, MaxPoolSize=값, CreationTimeout=값} 꼬리표 값을 사용한다. 최소 객체 수(MinPoolSize)는 COM+ 애플리케이션 시작과 함께 풀에 생성될 컴포넌트 인스턴스의 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 COM+ 런타임은 새로운 컴포넌트의 인스턴스를 적시 활성화시킨다. 즉, 적시 활성화가 사용되면 클라이언트가 사용하는 컴포넌트는 메소드를 호출할 때마다 달라진다. 적시 활성화는 트랜잭션의 정확함, 특히 일관성과 격리성을 지키는데 도움을 준다. 퍼베이시브 서비스를 PSM에 명세한 예는 그림 3과 같다.

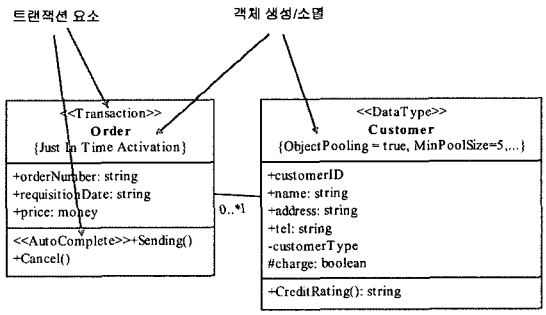

그림 3 PSM에 퍼베이시브 서비스를 명세한 예

4. PS-.NET 프로파일

본 장에서는 퍼베이시브 서비스 명세를 지원하는 .NET/C# 용 UML 프로파일을 제안한다. 이 프로파일은 .NET/C# 관련 UML 프로파일을 가정하고 여기에 본 논문에서 정의한 퍼베이시브 서비스 요소를 추가한 것으로 본 논문에서는 이를 "PS-.NET profile"이라 정의한다.

4.1 PS-.NET 프로파일 메타모델

그림 4에서 구성 요소 별로 살펴보면 먼저 Profile은

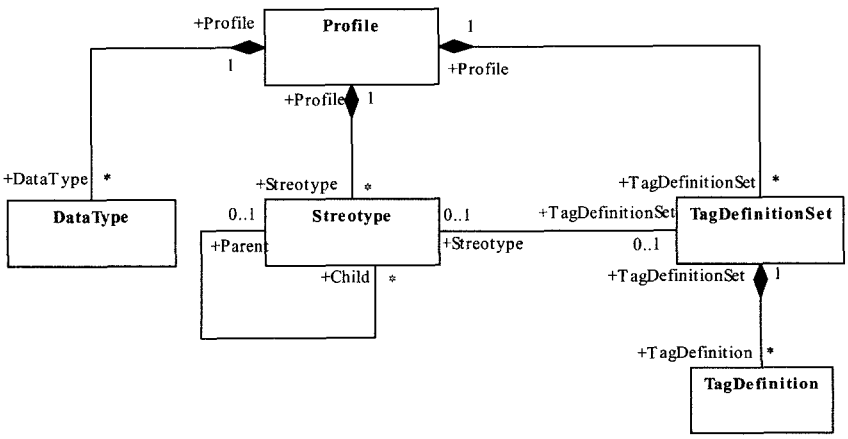

그림 4 PS-.NET Profile의 메타모델

여러 개의 Stereotype, Data Type, TagDefinitionSet들을 포함한다. Stereotype은 UML의 특정 요소들을 확장하는 것으로 하나의 Stereotype 아이콘이 존재하거나 없을 수 있다. TagDefinitionSet은 특정 요소의 추가적인 정보들을 추가하기 위한 단위이다. 이것은 여러 개의 TagDefinition들로 구성되며 어떤 요소들에 대해 적용될 수 있는지에 대한 정보를 나타낸다. 이것은 특정 Stereotype에 연결될 수 있는데, 이런 경우 Stereotype의 추가적인 속성들로써 이해될 수 있다. TagDefinition은 TagDefinitionSet의 구성 요소이며 하나의 확장 정보를 의미한다. Data Type은 프로파일이 포함하는 기본 데이타타입을 표현한다. PS-.NET Profile은 OMG의 UML Profile의 기능을 그대로 지원한다. 또한 MOF를 준수하므로, MOF를 준수한 UML 도구 및 MDA 도구에서 PS-.NET Profile의 사용이 가능하다.

4.2 PS-.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 메타모델 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 PS-.NET profile에서 .NET/C# 명세를 위한 UML 프로파일의 구성요소들을 정의한 것이다. 이들 프로파일 요소는 .NET/C# 용 PSM 모델링에 사용되고, 나아가 이렇게 명세한 내용들은 모델 변환을 통하여 .NET/C# 코드로 매핑이 되며, 생성된 코드는 COM+ 컴포넌트 용 닷넷 클래스가 된다.

4.3 PS-.NET 프로파일 문서 내용

앞에서 정의한 PS-.NET Profile의 메타모델과 구성요소들을 실제 적용해 XMI/XML 기반의 프로파일 문서를 작성하였을 경우 프로파일의 일부 내용을 살펴보면 아래 그림 5와 같다. 이 프로파일은 StarUML에 적용하여 사용할 수 있도록 아이콘 파일(.ico)을 만들어 StarUML 프로젝트에 포함시킨 후 사용하게 된다.

4.4 PS-.NET 프로파일을 이용한 변환 과정

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

표 3 .NET용 PSM 메타모델 정의서 항목

계층	PSM 메타 모델 요소	유형	정의	메타클래스
Presentation	ASPX	Stereotype	사용자 인터페이스 담당	UML Class
Business	DotNet Assembly	Stereotype	재사용 가능하며 버전관리가 가능한 컴포넌트	UML Component
Persistent	Data Type	Stereotype	정보가 저장되어져 있는 데이타셋 클래스	UML Class
Common	C# Operator	Stereotype	C# Operator 지정자	UML Operation
	...	...	...	...

표 4 .NET/C# 명세를 위한 UML 프로파일 요소

요 소	명세 표기법	정 의	적용되는 UML 요소
CSharp SourceFile	«CSharpSourceFile»	C# 구현 코드가 들어가 있는 소스 파일	UML Component
DotNet Assembly	«DotNetAssembly»	C# 소스 파일의 컴파일 결과로 얻어지는 닷넷 어셈블리	UML Component
CSharp Delegate	«CSharpDelegate»	C#의 델리게이트(delegate)임를 표시	UML Class
CSharp Struct	«CSharpStruct»	C#의 구조체(struct) 타입	UML Class
CSharp Event	«CSharpEvent»	C#의 이벤트 객체를 정의할 때 사용	UML Operation
CSharp Property	«CSharpProperty»	C#에서 클래스나 구조체의 속성을 나타내는 Property임을 나타냄	UML Operation
CSharp Operator	«CSharpOperator»	C#에서 연산자(operator)를 오버로드하는 연산자 함수(Operator Function)임을 표시. Operation의 이름이 오버로드 되는 연산자	UML Operation
CSharp Indexer	«CSharpIndexer»	C#에서 클래스나 구조체를 일반 배열과 같이 접근할 수 있도록 해주는 Indexer임을 나타냄	UML Operation
ASPX	«ASPX»	닷넷의 웹 페이지 확장자 명으로 클라이언트측 웹 페이지임을 나타냄	UML Class
Data Type	«DataType»	정보가 저장될 객체로 데이타베이스에 매핑시 테이블로 대응이 됨	UML Class
...	...	...	...

```
<?xml version="1.0" encoding="EUC-KR" ?>
<PROFILE version="1.0">
 <HEADER>
  <NAME>PS-.net</NAME>
  <DISPLAYNAME>PS-.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>Pervasive 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>
```

그림 5 PS-.NET Profile 문서

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 앞에서 정의하고 작성한 PS-.NET Profile과 StarUML, 그리고 StarUML 기반의 애드인을 이용하여 PSM 모델을 소스 코드로 자동 생성하는 과정을 제안한다. 먼저 앞에서 정의한 대로 PS-.NET Profile을 XML/XMI 형식으로 작성한 후 이것에 대한 아이콘 파일을 만들어 StarUML에서 사용할 수 있도록 한다. 이후 퍼베이시브 서비스에 대한 모델 변환을 위한 규칙을 변환 정의 언어를 이용하여 구현 하고, 이를 StarUML용 애드인으로 개발한다. 변환규칙을 구현한 예는 5장 사례연구에서 자세히 소개할 것이다. 마지막으로 PS-.NET Profile에 정의된 내용에 따라 .NET/C# 용 PSM을 설계하고, 이를 툴과 애드인을 이용하여 소스 코드를 생성하게 된다. 이를 그림으로 나타내면 그림 6과 같다.

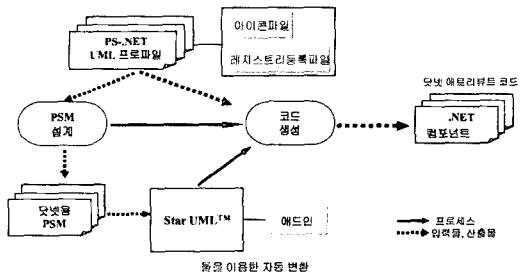

그림 6 PS-.NET 프로파일을 이용한 소스 코드 생성 과정

5. 사례연구

본 논문에서 적용 사례는 기업 정보 시스템의 영업관리 모듈 중 주문관리 기능에 대하여 퍼베이시브 서비스가 필요한 클래스에 대하여 적용하였다. 이는 Windows XP, StarUML, .NET/C#, and XML/XMI를 기반으로 구현 하였다.

5.1 PS-.NET 프로파일을 적용한 PSM 설계

먼저 StarUML에서 새로운 프로젝트를 생성한 후 그림 7과 같이 앞에서 작성한 PS-.NET Profile을 포함시킨다.

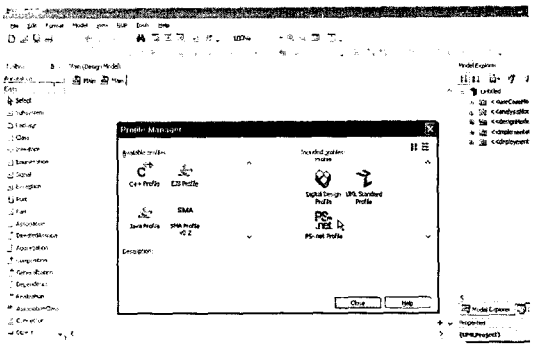

그림 7 작성한 PS-.NET Profile을 프로젝트에 포함시킨 모습

PS-.NET profile을 포함시킨 후에는 개발하려는 시스템에 대한 PSM을 설계한다. 이때 앞에서 정의한 .NET/C# 용 명세 요소를 사용한다. 그런 후 그림 8과 같이 작성한 닷넷용 PSM에 PS-.NET profile에 정의되어 있는 퍼베이시브 서비스요소를 필요한 클래스나 컴포넌트 혹은 메소드에 명세 한다.

그림 8은 PS-.NET profile의 스테레오 타입 항목의 내용을 명세한 경우로 'Order'클래스가 트랜잭션 속성을 지원하며 'Sending' 메소드는 트랜잭션 종료에 대한 속성이 'AutoComplete' 속성을 지원하도록 명세 하였다.

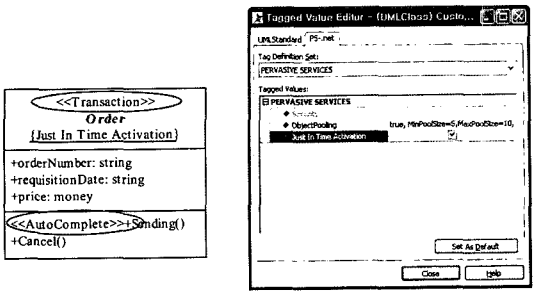

그림 8 PS-.NET Profile에 정의되어 있는 퍼베이시브 서비스 요소를 명세한 모습

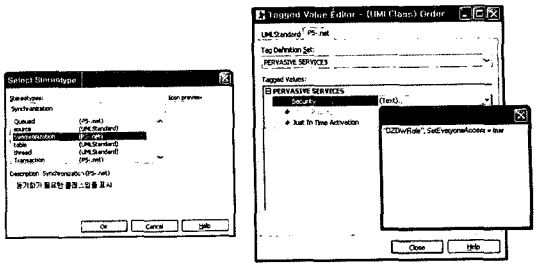

그림 9 확장 속성 편집기를 이용하여 퍼베이시브 서비스 요소를 명세 하는 모습

이렇게 스테레오 타입의 명세 외에 태그정의-집합의 태그정의 항목의 내용을 추가할 수 있다. 태그정의 항목은 앞에서 살펴본 바와 같이 프로파일 문서에 해당 항목을 XML 태그로 정의하면 오픈 API를 통해 아래 그림 9와 같이 확장 속성 편집기에 해당 속성이 추가되고 이를 이용해서 속성값을 입력한다.

이렇게 명세한 내용들은 모델 변환을 통하여 닷넷 애트리뷰트 코드로 매핑이 되며, 생성된 코드는 퍼베이시브 서비스를 지원하는 COM+ 컴포넌트 용 닷넷 클래스가 된다.

엔터프라이즈 시스템의 영업관리 모듈 중 주문관리 기능에 대한 클래스 다이어그램을 모델링 한 후 닷넷에 맞게 보다 상세한 PSM을 모델링 한 결과는 그림 10과 같다.

5.2 소스코드 변환

퍼베이시브 서비스 항목들과 함께 닷넷에 맞게 보다 상세하고 구체적으로 PS-.NET Profile의 요소들 을 이용하여 PSM을 완성하게 되면 개발한 애드인을 통하여 자동으로 닷넷용 소스코드가 생성된다. 애드인 개발에 앞서 PSM에서 코드로의 변환을 위하여 필요한 변환 규칙들과 제약조건을 정의한다. 각각의 변환규칙은 퍼베이

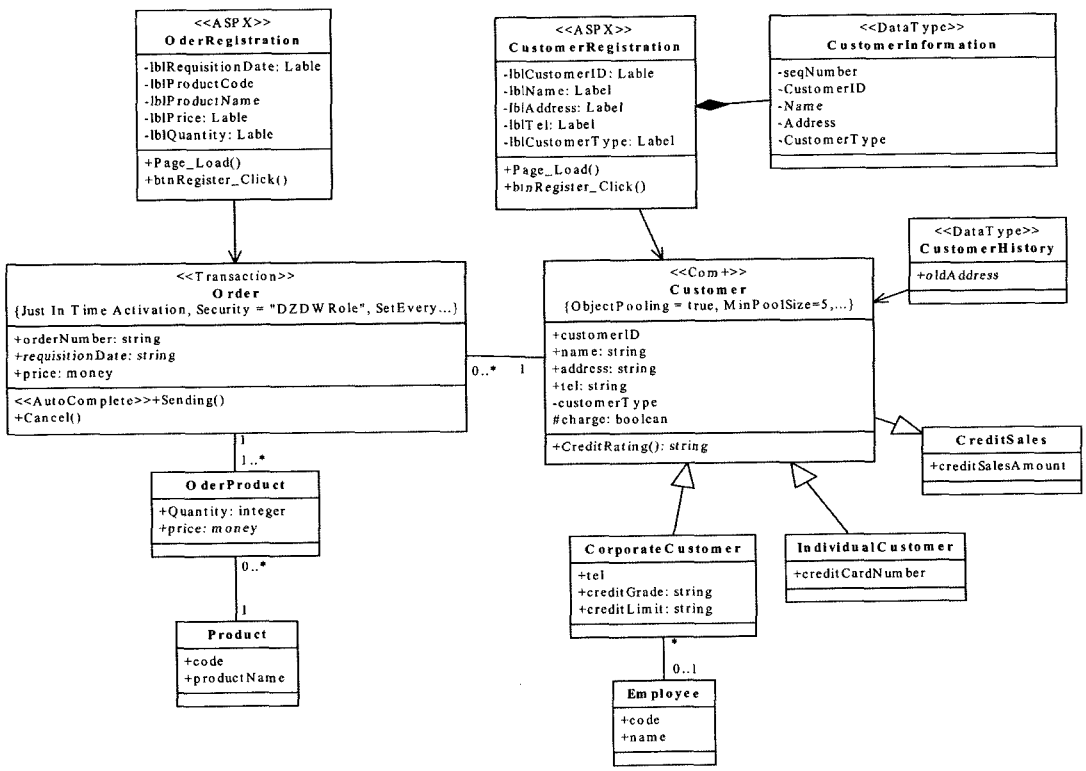

그림 10 주문관리 클래스 다이어그램(닷넷용 PSM)

표 5 변환규칙 정의서 항목

PSM 메타 모델 요소	변환 유형	변환 규칙	제약 조건
PSM 모델 구성항목	클래스, 오퍼레이션, 어트리뷰트 생성 등	모델 변환 시에 적용되는 변환 규칙 기술	변환규칙에 필요한 제약 조건을 기술
...	...	...	...

시브 서비스와 PSM 메타모델 정의서에 작성된 각 요소 간의 변환 관계를 정의함으로써 작성된다. 표 5와 같이 변환규칙 정의서는 각 메타모델 구성요소와 변환유형, 변환규칙, 제약조건으로 구성된다.

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[13]. 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 11은 QVT를 기반으로 PSM모델을 닷넷 코드로 매핑할 때 사용되는 정의인데 UML Profile에 정의된 퍼베이시브 서비스 요소는 닷넷 애트리뷰트 클래스로

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

  S := GetStringTaggedValue(AClass, 'PS-.net', 'PERVASIVE 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, 'PS-.net', 'PERVASIVE 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;
  ...
```

그림 11 변환 정의 언어를 이용한 변환규칙 구현 및 애드인 소스코드

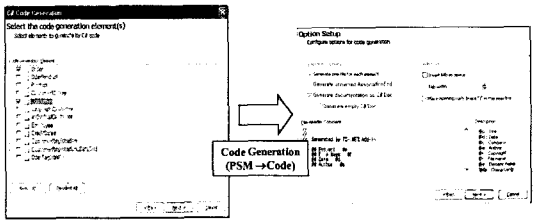

그림 12 소스코드를 생성할 요소 및 옵션 선택

변환시키고, 퍼베이시브 서비스 요소에서 사용한 인수 또한 닷넷 애트리뷰트 클래스의 인수로 변환시키겠다는 정의를 하고 있다. 또한 이를 StarUML 기반의 애드인으로 구현한 소스코드의 예를 보여주고 있다. 소스/타겟 조건은 OCL Boolean 표현식으로 작성한다[16].

그림 12는 개발한 애드인을 이용하여 앞서 명세한 퍼베이시브 서비스 요소를 포함한 .NET/C# 코드를 생성하는 모습이다. 작성한 닷넷용 PSM에서 생성할 클래스를 선택하고 생성 옵션을 설정한 후 모델을 변환 과정을 통해 실행하면 최종 작성 결과인 코드가 생성되게 된다.

그림 12의 모델 변환 과정을 통해 실행하면 최종 작성 결과인 코드는 그림 13, 14와 같다. 그림 13에서 보다시피 코드 'Order' 클래스 위에 트랜잭션지원과 객체 생명주기(적시활성화)요소에 관한 닷넷 애트리뷰트 코드가 생성 되었고 'Sending' 메소드위에 트랜잭션 종료 요소에 관한 코드가 생성되었음을 알 수 있다.

그림 14는 'Customer' 클래스에 대해서 객체 풀링 속

```
//
[Transaction(TransactionOption.Supported)]
[JustInTimeActivation(true)]
[SecurityRole("DZDWRole", SetEveryoneAccess = true)]
public class Order {
        string orderNumber ;
        string requisitionDate ;
        money price ;

        public Order()
        {
                ...
        }

        [AutoComplete]
        public void Sending()
        {
...
```

그림 13 변환을 통해 작성된 "Order" 클래스

```
//  Generated by PS-.NET Add-In
//  @ File Name : Customer.cs
//  @ Author :

[ComVisible(true)]
[ObjectPooling(true, MinPoolSize=5,MaxPoolSize=10,
CreationTimeout=5000)]

public class Customer : CreditSales{
        public object customerID ;
        public string name ;
        public string address ;
        public string tel ;
        private object customerType ;
        protected boolean charge ;
        public string CreditRating(){
            }
}
...
```

그림 14 변환을 통해 작성된 "Customer" 클래스

성의 값을 각각 지정하여 설정한 요소에 관한 닷넷 애트리뷰트 코드와 COM 관련 애트리뷰트 코드가 생성되었음을 알 수 있다.

6. 평 가

기존 연구와 비교해 보면, Kath의 Executable Models [10]는 오픈 모델링 기반구조에 MOF가 근간이 되는 기초 기술로 사용되며, 이 기반 구조를 사용하여 PSM을 명세 한 후 이를 모델 변환기에서 각각의 플랫폼에 맞는 코드를 생성하는 프로세스를 제안하고 있다. 하지만 제안된 프로세스가 추상적이고, 생성된 코드의 구체적인 형태나 방법은 언급하지 않았으며, 엔터프라이즈 시스템 개발에 필수적인 기능인 퍼베이시브 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 소스 코드를 생성한다. 하지만 구체적인 프로파일이나 OCL등을 적용한 UML 확장 장치와 모델링에 대한 언급이 없으며, 퍼베이시브 서비스에 대한 모델 변환은 일부만 언급하였다.

본 논문에서는 퍼베이시스 서비스와 .NET/C# 명세를 위한 요소들을 제안하고, 이 요소들을 PSM 메타모델을 준수하는 UML 프로파일로 정의하여 PSM에 명세할 수 있는 장치를 만들었다. 또한, 이를 변환규칙을 구현한 툴로 개발함으로써 퍼베이시브 서비스 기능을 지원하는 닷넷 컴포넌트를 자동 생산할 수 있게 하였다.

본 논문에서 제안한 기법을 평가하기 위하여 다음과 같은 평가항목을 선정하였다. 평가 항목의 선정 기준은 OMG의 MDA 가이드 1.0[1] 준수 여부, OMG의 MOF, XMI, QVT 등 표준의 준수 및 사용 여부와 닷넷 플랫폼 및 퍼베이시브 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: PSM 모델 명세 및 소스 코드로의 변환을 위한 XML/XMI 기반의 UML Profile을 제공하는지에 대한 평가
- .NET/ C# 용 UML Profile의 지원 여부: .NET/ C# 용 PSM 모델 명세 및 소스 코드로의 변환을 위한 UML Profile을 지원하는지에 대한 평가
- 퍼베이시브 서비스 요소 명세 및 이를 지원하는 코드 생성 여부: 퍼베이시브 서비스를 명세하기 위한 장치 및 이를 지원하는 코드 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 적용성을 향상시킬 수 있는지를 평가

이상의 평가 항목으로 표 6에서 Kath의 Executable Models와 Oldevik의 Model to Text Transformations 모델을 대상으로 본 기법을 비교해 보았다.

표 6 기존 연구와의 비교 평가

O:지원, △:부분 지원

평가 항목 \ 대상	Kath의 Executable Models[10]	Oldevik [11]	본 논문의 기법
OMG 표준인 MDA 가이드 1.0 준수	O	O	O
OMG 표준인 MOF, XMI, QVT 준수 및 사용	△	O	O
UML Profile의 제공			O
.NET/ C# 용 UML Profile의 지원			O
퍼베이시브 서비스 요소 명세 및 이를 지원하는 코드 생성		△	O
도구의 제공	O		O
기법에 대한 상세지침 제공			O
적용 예제 제공			O

7. 결 론

본 논문은 PSM에서 소스코드를 생성하는 방안에 대한 연구로써 현재까지 발표된 UML 프로파일의 경우 PIM에서 PSM을 생성 하거나 PSM에서 소스코드를 생성하였을 시 퍼베이시브 서비스에 대한 명세 및 모델 변환을 위한 장치가 없다는 점에서 출발하였다. 현재까지의 MDA 연구는 구조적 모델의 변환은 성공적으로 이루어지고 있으나 생성된 소스 코드는 엔터프라이즈 애플리케이션에서 필요한 퍼베이시브 서비스는 지원하지 않고 있으며, 비 효율적인 면이 여럿 남아있다.

본 논문에서는 엔터프라이즈 애플리케이션에서 필수적인 트랜잭션, 보안, 동기화 서비스, 객체 풀링 등의 퍼베이시브 서비스에 대한 소스코드 자동 생성을 위하여 PS-.NET profile을 제안 한다. 또한 이를 오픈 소스 모델링 플랫폼인 StarUML에 적용할 수 있도록 구성하고, PS-.NET profile의 정의에 따라 닷넷용 PSM을 설계하였으며, 소스코드 생성을 위한 애드인을 개발하여 이를 소스코드로 자동 생성하여 보았다. 5장의 사례연구에서는 단 두 개의 클래스에 대해 적용해 본 결과 기존 순공학 툴에 비해 퍼베이시브 서비스에 대하여 5 개 라인이 더 자동 생성되었지만, 예를 들어 200개의 클래스가 존재하는 큰 규모의 시스템일 경우 퍼베이시브 서비스 기능이 필요한 클래스가 150개라 가정할 경우 최소 150라인에서 많게는 1500라인이 훨씬 넘는 코드를 자동 생성함으로써 높은 개발 생산성을 보일 것이다. PS-.NET profile은 XMI 기반의 XML문서로서 쉽게 내용을 추가하거나 수정할 수 있으며, StarUML과 개발한 애드인은 외부 API를 이용하여 쉽게 이를 적용할 수 있다. 또한 PS-.NET profile은 OMG의 UML Profile의 기능을 그대로 지원하며, MOF를 준수하므로 MOF를 준수한 UML 도구 및 MDA 도구에서 사용이 가능하다. 이처럼 본 논문에서 제안한 방법을 사용할 경우 퍼베이시브 서비스 기능을 지원하는 닷넷용 소스 코드를 쉽게 자동 생성 할 수 있으며 높은 개발 생산성, 확장성, 이식성 및 유지보수성을 증가 시킬 수 있다.

참 고 문 헌

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

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

[3] David S. Platt, *Introducing Microsoft .NET Second Edition*, Microsoft Press, 2002.

[4] 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.

[5] Plastic Software Inc, http://www.StarUML.com.

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

[7] OMG, "UML™ Profile and Interchange Models for Enterprise Application Integration(EAI) Specification," 2002.

[8] OMG, "UML Profile for CORBA Specification V1.0, OMG," Nov. 2000.

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

[10] Kath, O., Blazarenas, A. and Funabashi, M., "Towards Executable Models: Transforming EDOC Behavior Models to CORBA and BPEL," Proceedings of the 8th IEEE Intl Enterprise Distributed Object Computing Conference (EDOC 2004) 2004.

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

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

[14] Lowy, Juval. Programming .NET Components. O'Reilly, 2003. 459 p.

[15] The Microsoft Developer Network (MSDN). Available at http://msdn.microsoft.com/ library/default.asp

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

[17] Jana Koehler, Rainer Hauser, Shubir Kapoor, Fred Y. Wu, Santhosh Kumaran, "A Model-Driven Transformation Method," Proceedings of the Seventh IEEE International Enterprise Distributed Object Computing Conference (EDOC'03) 2003.

[18] OMG, *UML Profile for EDOC V1.0*, http://www. omg.org /technology/documents/formal/edoc.htm, 2004.

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

[20] OMG, *UML Profile for Meta Object Facility V1.0*, 2004.

금 득 규

1995년 강남대학교 산업공학과 공학사 2005년 숭실대학교 정보통신학과 공학석사. 2006년~현재 숭실대학교 컴퓨터학과 박사과정. 2000년~2001년 (주)지오매니아 연구소 연구원. 2001년~2003년 (주)더존디지털웨어 연구소 주임연구원. 2003년~2004년 (주)필컴정보시스템 연구소 책임연구원. 관심분야는 컴포넌트 기반 개발(CBD), 모델 기반 아키텍처(MDA), 서비스지향 아키텍처(SOA)

김 수 동

정보과학회논문지 : 소프트웨어 및 응용
제 34 권 제 4 호 참조