메타데이터를 이용한 응용프로그램 생성기의 개발

김 치 수†

요 약

일반적으로 소프트웨어 개발 과정은 요구사항 분석, 설계, 코딩, 테스트, 유지보수의 과정으로 이루어진다. 그러나 개발 과정에 있어 설계의 빈번한 수정은 그 다음 단계인 코딩을 매우 어렵게 만든다. 즉, 설계와 구현 사이에 항상 불일치를 유발하게 된다.

본 논문에서는 시스템 설계와 구현 사이의 불일치를 줄이고, 소프트웨어의 개발이 신속하고 유연하게 되도록 비즈니스 로직을 식별하여 응용프로그램을 생성시켜주는 도구를 개발 하였다. 또한 비 프로그램 기반 응용 프로그램 시스템 방법도 제안 하였다. 이 방법은 동적 메소드를 통하여 실행 시간에 시스템 설계의 메타데이터를 편집해서 실제로 응용프로그램을 구축하거나 수정한다.

키워드 : 메타데이터, 응용프로그램, 생성기, 설계

The Development of the Application Program Generator based on Meta-Data

Kim Chi Su†

ABSTRACT

Generally, a software development process is composed with requirements analysis, design, coding, test and maintenance.

However, some changes of the design step are difficult to complicate the next step in the development process. It always causes the disagreement between design and implementation step.

In this paper, we have developed a tool which can generate an application program. The tool can reduce the disagreement between system design and implementation and recognize the business logic to develop the software rapidly and flexibly.

In addition, we proposed a non-program-based application program system approach was proposed, too. We can generate and modify an application program with this method which can edit the meta data of a system design by the dynamic method for the execution time.

Key Words : Meta-Data, Application Program, Generator, Design

1. 서 론

소프트웨어 개발에 있어 프로그래밍의 어려움과 다양한 사용자 요구사항의 빈번한 변화로 인한 시스템 설계와 구현 사이의 문제는 항상 있어 왔다. 많은 UML 기반의 분석/설계 도구가 개발되어 생산성 향상에 기여하고 있고, 코드 생성 기능 등을 이용하여 설계와 구현 사이의 문제에 도움을 주고는 있지만 설계 모델이 변경되거나 수정되면 새로운 코드를 생성하여 적용하거나 개발자가 수작업을 통하여 일일이 코드를 수정해야 한다[1-3].

따라서 본 논문에서는 설계와 구현사이의 문제점을 해결하기 위하여 비즈니스 객체 모델을 메타모델로 변환하는 코드 생성 방법을 제안하고, 이 메타모델로부터 구현코드를 생성시켜주는 도구를 개발하였다.

또한 이 도구를 개발하기 위해 사용된 두 번째 방법의 비 프로그램 기반 응용 프로그램 시스템은 개발자만이 시스템 소프트웨어를 수정하거나 구현하던 것을 최종 사용자에게 유연성을 증가시켜 소프트웨어 개발에서 시스템 설계와 구현사이의 문제점을 실행 시간(run-time)에 바꿀 수 있도록 사용자에게 초점을 맞춘 것이다.

이 방법은 환경의 변화나 요구사항에 맞도록 애플리케이션을 다시 작성할 경우 또는 같은 영역의 시스템 개발 시에 메타데이터로 저장된 객체모델을 사용함으로써 소프트웨어 개발의 생산성 향상, 개발 시간의 단축, 그리고 좀 더 쉽고 유연하게 소프트웨어를 개발할 수 있다.

본 논문에서 객체지향 분석과 설계를 통해서 정의된 객체는 표준 비즈니스 객체 모델을 시스템의 GUI설계와 함께 지속적인 메타데이터로 저장하고 객체지향 프로그래밍 언어와 객체지향 데이터베이스로 모델에 변경 없이 시스템으로 바로 구현되게 된다. 시스템의 변화는 간단히 시스템의 메타데이터를 편집함으로써 새로운 코드가 생성된다. 그 결과 시스템 분석/설계의 산출물을 직접적인 구현으로 사용함으

※ 이 논문은 교육인적자원부의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(R05-2003-000-11824-0).

† 종신회원 : 공주대학교 컴퓨터공학부 교수(공학연구원)

논문접수 : 2005년 11월 17일, 심사완료 : 2005년 12월 27일

로써 시스템 분석/설계와 구현 사이의 문제점을 해결하는데 사용한다.

또한 시스템 설계를 지속적인 메타데이터 형태로 저장함으로써 기존의 다이어그램이 단지 시스템 모델링을 표현할 수 있는 산출물인데 반해, 메타데이터는 시스템 GUI 설계 정보, 비즈니스 로직 등 다른 정보들을 보다 상세하게 포함할 수 있으며, 시스템 설계를 편집할 수 있어서 개발자와 최종 사용자는 시스템 설계의 메타데이터를 편집함으로써 요구사항이 바뀔 때 시스템 설계에 변화를 다양한 목적으로 직접적으로 처리할 수 있다는 장점을 가지고 있다.

2. 관련 연구

메타모델링의 중요성에도 불구하고, 메타모델은 부족한 면을 많이 가지고 있다[4-6]. 실례로 메타모델의 본질과 구조에 대한 정의가 거의 없다.

본 논문에서는 메타 모델링을 다양한 시스템 설계 모델을 정의하고 저장하는 전반적인 개념으로 사용하였다.

메타데이터에 대한 가장 단순하면서도 유용한 정의는 "데이터에 대한 구조화된 데이터"이다. 그리고 데이터나 작업에 대한 속성이나 내용을 기술한다.

본 논문에서는 시스템 설계 정보를 영구적인 데이터로 저장하기 위해서 메타데이터를 사용한다.

Rational Rose[7, 8]는 거의 대부분의 UML의 주요특징을 지원하는 툴로 사용자에게 객체지향 분석, 설계, 모델링, 구현의 기능을 제공한다.

본 논문에서는 Rational Rose를 사용해서 비즈니스 애플리케이션 생성기 UML 모델링, 비즈니스 객체모델, 클래스 메타모델, GUI 메타모델 등을 표현한다.

JBuilder는 Borland Software Corporation에서 만든 케이스 도구로 앞서가는 가장 포괄적인 시각화 반복 개발 환경(IDE : Interactive Development Environment)으로 자바 애플리케이션을 구축하기 위한 도구의 하나로 알려졌다[9, 10]. 본 논문에서는 케이스 도구을 개발하는데 있어 Borland JBuilder를 주로 사용하였다.

이 툴들은 소프트웨어 개발을 향상시키는데 큰 기여를 했지만 특별한 목적으로만 사용되기 때문에 여전히 설계와 구현 사이의 문제점을 해결할 수는 없다는 것이다. Borland JBuilder의 경우 코드 편집과 시각화 정의에 의한 GUI 윈도우 생성은 강력하지만 시스템 모델링은 다룰 수 없다. 이와 비교해 볼 때 Rational Rose는 시스템 모델링엔 좋지만 GUI 환경을 다룰 수 없는 문제점이 있다.

본 논문에서는 이러한 문제점을 자원의 검색, 관리, 내용등급, 보존, 저작권 관리, 전자상거래 등에 다양하게 활용될 수 있는 메타데이터[11]를 활용하여 시스템 모델을 구축하는 사례로 GUI 윈도우 생성과 시스템 모델링을 양쪽 다 고려하여 설계와 구현사이의 문제점을 해결하려는 시도로써 메타데이터베이스를 케이스 도구에 효율적으로 활용하는 특징을 가지고 있다.

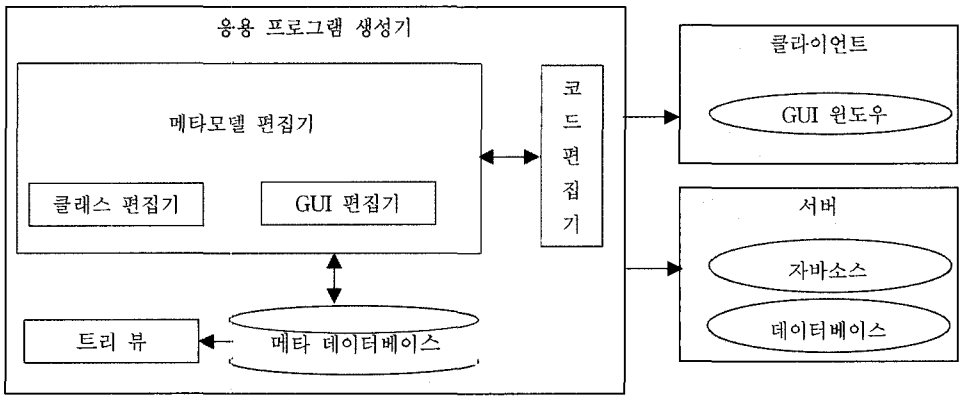
3. 응용 프로그램 생성기 설계

본 논문에서 개발한 툴은 (그림 1)과 같이 시스템 모델링, 시스템 GUI 설계, 코드생성, 데이터베이스 테이블 생성, GUI생성, 시스템 설계 편집 그리고 프로그램 코드 편집 등의 기능을 갖는 순수한 자바기반 통합개발환경으로 "응용 프로그램 생성기"라 명명하였다.

이 응용 프로그램 생성기는 다음과 같이 두 가지 핵심 부분으로 구성된다.

첫째는 시스템 설계에 대한 정보를 저장하는 데이터베이스로서 시스템 객체모델과 시스템 GUI 설계의 메타데이터를 저장하고 관리하는데 사용된다.

두 번째로 데이터베이스에 저장된 시스템 설계의 메타데이터는 애플리케이션에 대한 소스코드와 데이터베이스 테이블 생성을 지원하는데 사용한다. 또 응용 프로그램 생성기로부터 생성되는 소스코드는 서버 측 모델에 대한 모든 자바소스코드인 비즈니스 서비스 클래스, 클라이언트 측 GUI 윈도우 클래스, 애플리케이션의 데이터 저장을 위한 서버 측 데이터베이스 테이블, 다른 관련된 데이터베이스 관리 클래스들로 구성된다. 여기서 응용 프로그램 생성기의 GUI 편집기는 특별한 요구사항에 맞춰 시스템 설계를 하기 위한 메타데이터를 편집하는데 사용한다.



(그림 1) 시스템 전체 구성도

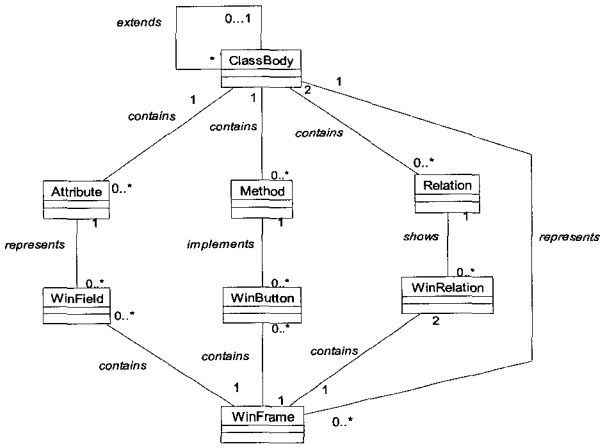
3.1 메타모델 설계

응용 프로그램 생성기는 기능성을 고려하여 두 부분의 메타모델로 설계된다. 하나는 시스템에 대한 시스템 데이터 모델을 기술하는데 사용되는 것으로 클래스 메타모델로 불린다. 다른 하나는 시스템의 객체 모델을 기반으로 한 시스템에 대한 GUI를 기술하는데 사용되는 GUI 메타모델이다.

클래스 메타모델은 시스템 설계의 객체 모델에 대한 전반적인 정보를 파악하기위해 ClassBody, Attribute, Method, Relation의 4개의 클래스로 구성된다.

GUI 메타모델은 시스템 객체 모델링을 기반으로 시스템 GUI를 설계하는 전반적인 정보를 파악하기위해 WinFrame, WinField, WinButton, WinRelation의 4개의 클래스로 구성된다.

본 논문의 응용 프로그램 생성기에서는 클래스 메타모델과 GUI 메타모델을 각각 서로 독립적이지 않고 (그림 2)와 같이 응용 프로그램 생성기의 완전한 메타모델로 함께 구성되어 사용된다.



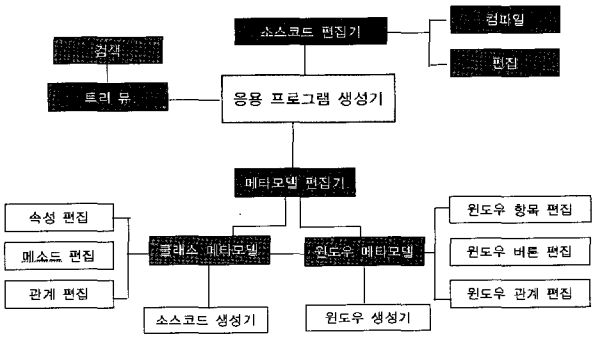
(그림 2) 응용 프로그램 생성기의 완성된 메타모델

3.2 응용 프로그램 생성기의 기능과 구조

응용 프로그램 생성기는 순수한 자바기반으로 설계된 것으로 시스템 모델링에서 자바코드 편집까지 수행한다. (그림 3)은 응용 프로그램 생성기의 주요 기능을 나타내 주고 있는 것으로 응용 프로그램 생성기의 핵심 모듈은 트리 뷰, 모델 편집기, 소스코드 편집기로 구성된다.

트리 뷰는 사용자가 시스템에 대한 유용한 정보와 시스템에 이미 저장되어 있는 관련된 클래스를 빨리 찾을 수 있도록 설계되었다. 이것을 수행했을 때 트리구조를 보여주며 기존에 있는 모든 기본 정보와 시스템에 정의된 클래스에 따른 속성, 메소드, 클래스사이의 관계를 보여준다.

메타모델 편집기는 응용 프로그램 생성기의 주요 특징으로 시스템 설계의 메타데이터를 편집하고 저장하며 애플리케이션에 대한 코드를 생성하도록 설계되어져 있고, 클래스 메타모델 편집과 GUI 메타모델 편집 두 부분으로 나뉘어져 있다. 클래스 메타모델 편집은 객체 모델링과 관련된 모든 프로세스를 다루는데 사용되고 GUI 메타모델 편집은 시스



(그림 3) 응용 프로그램 생성기의 기능

템 GUI 배치에 관련된 프로세스를 다루는데 사용된다.

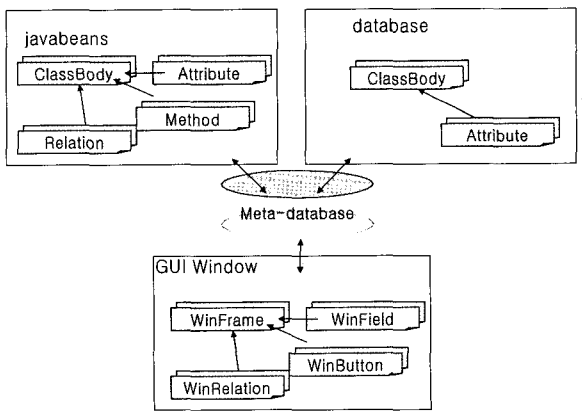
소스코드 편집기는 사용자가 시스템에 의해 생성된 자바 소스코드를 편집, 컴파일, 실행하는 기능을 제공한다.

또한 클래스 바디 편집기는 시스템 객체 모델인 클래스 메타 모델과 GUI설계의 메타 모델을 편집한다. DBConnection은 응용 프로그램 생성기에서 JDBC를 사용해서 자바 클래스와 데이터베이스 연결을 만들어주고, SQL쿼리를 수행한다.

3.4 코드 생성 설계

소스코드의 구현은 우선 설계 단계의 산출물인 UML의 클래스 다이어그램을 메타데이터 다루게 되는데 (그림 4)에서처럼 클래스명은 ClassBody 객체로, 클래스의 속성은 Attribute 객체로, 클래스의 메소드는 Method 객체로, 클래스의 관계는 Relation 객체로 추출되어 메타데이터 테이블에 저장한다.

저장된 타데이터는 소스코드로 직접 매핑되고, ClassBody 객체는 서버측 자바 빈즈 클래스로 매핑된다. Attribute 객체는 자바 빈즈 클래스의 속성으로 getter/setter 메소드와 함께 매핑된다. Method 객체는 자바 빈즈 클래스의 메소드로 매핑된다. 자바 빈즈 클래스에서 ContainsOne 타입을 가지고 있는 Relation 객체는 Relation 클래스에서 관련된 클래스에 의해 정의된 속성으로 관련된 객체에 매핑된다. 자



(그림 4) 코드생성 흐름도

바 빈즈 클래스에서 ContainsMany 타입을 가지고 있는 Relation 객체는 Vector type 객체 속성으로 매핑된다.

ClassBody는 또한 애플리케이션을 생성하기 위한 서버 측 데이터베이스 테이블로 대응된다. 데이터베이스 테이블에서 Attribute 객체는 테이블의 항목으로 매핑된다.

WinFrame 객체는 클라이언트 측 자바 클래스로 매핑 되는데 JFrame, JDialog 클래스의 확장이다. 윈도우에서 WinField 객체는 TextField, JTextArea, JComboBox, JLabel을 가지고 있는 JCheckBox로 적절히 매핑된다.

윈도우에서 WinButton 객체는 JButton 으로 매핑된다. "TabbedPane"타입의 뷰를 가지고 있는 WinRelation 객체는 하나의 JFrame에 있는 하나의 탭 패인으로서 두개의 데이터 입력 패인으로 매핑된다. "List" 타입의 뷰를 가지고 있는 WinRelation 객체는 데이터 입력 윈도우에서 리스트의 항목을 처리하는데 사용되는 "nested button pane"과 함께 탭 패인 안에 객체들과 관련된 리스트와 매핑한다. 데이터 입력 윈도우에서 "Table"의 뷰 타입을 가지고 있는 WinRelation 객체는 관련된 객체의 테이블을 포함하는 탭 패인과 매핑한다.

(그림 5)는 일반적인 판매주문 시스템에서 Customer 클래스를 가지고 생성된 자바 소스 코드의 예를 보여준다.

```
import java.util.*;
/* Customer data */
public class Customer implements java.io.Serializable
//=== Attributes ===
Public String id; //null
public String name; //null
.....
public String homeAddress; //null
//=== constructors ===
public Customer(){
}
public Customer(String id, String name, String workPhone, String
H.P, String email, Address homeAddress, Address postAddress){
this.id=id;
.....
this.homeAddress=homeAddress;
this.postAddress=postAddress;
}
//=== Methods ===
public String getId(){
 return id;
}
public void setId(String id){
 this.id=id;
}
.....
public String toString(){
 retrun "Customer ID : "+id+"name";
}
}
```

(그림 5) 생성된 자바소스 코드의 예

4. 응용 프로그램 확장 기법

4.1 속성 변경

본 논문에서 제안하는 2번째 방법인 비프로그램 기반 응용 프로그램 시스템 설계에서는 실행 시간에 객체에 새로운 속성을 추가하거나 속성의 내용을 변경하기 위해 모든 속성을 해쉬 데이터 구조(hash data structure)를 사용해서 다룬. (그림 6)은 자바의 해쉬테이블에 클래스의 속성을 어떻게 저장하는가를 보여준다.

```
class AClass{
        private HashTable Properties=new HashTable();

        public void setProperty(String key, Objectvalue){
                properties.put(key, value);
        }
        public void getProperty(String key, Objectvalue){
                properties.put(key, value);
        }
```

(그림 6) 해쉬테이블에 클래스 속성 저장하기

이 코드에서 해쉬 테이블은 메타데이터의 속성과 동등한 키를 사용해서 속성을 인덱스한다. 해시테이블에 저장된 속성의 값은 총체적 데이터 타입(Object)으로 정의된다. 이것은 다른 타입(String or Integer)으로 타입캐스트(typecast)될 수 있다. setProperty 메소드는 클래스의 기존 속성을 갱신하거나 새로운 속성을 추가하는데 사용한다. getProperty 메소드는 속성값을 검색하는데 사용한다. 추가적으로 속성의 제거나 모든 속성값을 리턴하는 메소드는 클래스의 속성을 다루는 보조 클래스에 추가할 수 있다.

본 논문에서 클래스의 속성을 저장하는데 해쉬 테이블을 사용하면 사용자가 메타 데이터의 속성을 어떻게 나타내고 액세스하는 지에 대한 선택사항을 제공해서 코드의 복잡성을 데이터로 넘긴다. 또한 사용자에게 실행 시간에 동적으로 클래스의 속성을 변경할 수 있도록 허락한다. 따라서 소프트웨어의 유연성이 상당히 증가할 수 있다.

4.2 클래스의 변경

클래스 변경의 요구 시 (그림 7)에서 보여주는 바와 같이 기존방법인 우선 바인딩(early binding) 방법은 적재된 클래스를 실행 시간 까지 알 수 없기 때문에 사용할 수 없다. 본 논문에서는 이 문제를 해결하기 위해서 클래스의 지연바인딩(late binding)을 사용한다.

지연 바인딩은 알려지지 않은 클래스가 있을 때 시스템이 초기에 컴파일하고 적재한다. 또한 지연 바인딩은 Class type object의 선언을 필요로 하는데 알려지지 않은 클래스를 저장하기 위해 사용된다. Class type object는 파일 시스템에서 지연 바운드된 클래스를 위치시키기 위해 forName (classname) 메소드를 호출해서 메모리에 적재시키고 실행 시간에 프로그램 네임스페이스로 바인드한다. 그리고 지연

```
import AKnownClass;
.....
AKnownClass aClass= new AKnownClass();
System.out.println(aClass.toString());
```

(그림 7) 자바 클래스의 우선바이딩

```
.....
String ClassName="UnKnown";
Class aClass= ClassforName(className);
Object laterBinding=aClass.newInstance();

System.out.println(laterBinding.toString());
.....
```

(그림 8) 자바 클래스의 지연 바이딩

바인딩은 Object type object를 선언할 필요가 있는데 지연 바운드 된 클래스의 인스턴스를 저장하기 위한 것이다. 이 인스턴스는 지연 바운드된 클래스의 Class type object에서 newInstance()의 메소드의 호출을 통하여 얻어진다. 그런 다음 Object type object는 기본 타입의 하나로 타입캐스트될 수 있어서 다른 목적으로 사용될 수 있다.

지연바인딩은 우선 바인딩과 비교해볼 때 실행 시간에 클래스가 동적으로 변경되는 것을 허락하기 때문에 소프트웨어 개발에 있어 상당한 유연성을 가져온다. 또한 참조(reference)로만 사용되기 때문에 시스템이 더 많은 참조를 포함할 때, 더 큰 파일 사이즈와 더 긴 수행시간이 걸린다. 그리고 지연바인딩으로 구성한 시스템은 추상적이고 간결하며 시스템에서의 변경은 사용자에게 투명성을 제공한다. 그러나 추가적 오버로드가 실행 시간에 발생할 때 시스템이 느려지는 점과 시스템의 디버깅이 더 어렵다는 문제점도 갖고 있다.

4.3 코드 변경

비프로그램 기반 응용 프로그램 시스템에서 중요한 작업은 실행 시간에 메타데이터를 기반으로 생성되거나 변경되는 코드번역이다. 일반적 코드 생성은 소스코드를 쓰거나 고치고, 컴파일하고, 배치하고, 실행하는 단계를 통하여 변경된다. 이에 반해 비프로그램 기반 응용 프로그램 시스템에서 이 단계는 상당히 부적절하다. 이 문제를 해결하기 위한 방법으로 실행 시간에 코드 변경을 하기 위해 실행 가능한 인터페이스를 구현하는 클래스를 만드는 것이다.

실행 가능한 인터페이스는 활성화된 동안 실행되는 코드인 객체에 대한 공통 프로토콜을 제공하기 위해 설계한다. 실행 가능한 인터페이스는 run() 메소드로만 선언하는데 실행가능한 객체가 활성된 동안만 실행된다. run()메소드는 쓰레드 클래스를 서브클래스로 하고 run()메소드를 오버라이드하는 방법과 실행 가능한 인터페이스를 구현하는 클래스를 제공해서 run()메소드를 구현하는 방법이 있다. (그림 9)는 문자(letter technique)를 동적 코드로 어떻게 번역하는가를 보여준다.

실행 시간에 동적 코드 해석을 얻기 위해 첫 번째로 실행 가능한 인터페이스를 구현하는 클래스를 선언한다. 두 번째로 run() 메소드를 실행하는 줄을 추가한다. main() 메소드에서 run()메소드를 실행하는 줄은 프로그램을 실행을 시작하는데 사용된다. 세 번째로 클래스의 지연 바인딩 프로그램의 동적인 것을 수행하는·run() 메소드를 써서 실행 시간

```
.....
PrintWriter oStream;
.....
try{
        oStream=new printWriter(newFileOutStream(fileName));
}
catch(IOException){}
.....
//=========== 코드 생성 ============
oStream.println("JFrame aFrame = new JFrame(\" "+title+"\");");
oStream.println("JLabel aLabel = new JLabel(\" "+labelName+"\");");
.....
```

(그림 9) 소스코드를 작성을 이용한 코드 생성

```
public class GenericApplication implements Runable{
.....
        public static void main(String args[]){
        .....
        run();
}
.....
public void run(){
        .....
        //=========== 지연 바인딩 =======
        .....
        //=========== 코드번역 =========
        .....
        JFrame aFrame = new JFrame(aObject.getProperty("title"));
        JLabel aLabel = new JLabel(aObject.getProperty("label"));
        .....
        }
}
```

(그림 10) 실행 시간을 이용한 코드번역

에 코드를 수정하는 방식으로 해결할 수 있다. (그림 10)은 실행 시간을 이용한 코드 번역과정을 보여주고 있다.

5. 결론 및 향후 연구과제

시스템 설계의 빈번한 변화와 프로그래밍의 어려움으로 인한 설계와 구현사이의 문제점을 해결하기 위해 본 논문에서는 비즈니스 객체 설계의 산출물을 직접적인 비즈니스 객체 구현으로 사용하는 방법을 제안한다.

본 논문에서는 설계와 구현사이의 문제점을 해결하기 위해 비즈니스 애플리케이션을 핵심 비즈니스 로직과 화면표시 로직으로 나누어 식별하였다. 또한 이를 같은 시스템 설계에서도 공유할 수 있도록 지속적인 메타데이터로 저장한 후, 저장된 메타데이터를 직접적인 구현에 사용하였다. 따라서 요구사항에 맞는 애플리케이션을 구축할 수 있도록 함으로써 애플리케이션의 개발을 좀더 간단하고, 빠르며, 유연하게 할 것으로 기대된다.

향후 연구과제로 비즈니스 모델을 보다 더 총체적으로 파악하는 방법, 시스템에 대한 GUI를 생성하는 방법, 생성된 애플리케이션을 어떻게 보이게 할 것인가의 방법 등을 향상시키기 위한 구체적인 연구가 필요하며, 완전한 코드생성 방법의 연구가 계속되어야 한다.

참 고 문 헌

[1] Bradford, K., "Software Components as Application Building Blocks" at URL : http://www.quoininc.com/quoininc/ComponentsABB. html, 1998.

[2] Chappel, D., "The Next Wave: Componet Software Enters the Mainstream," at URL : http://www.rational.com/uml/resourcepapers, 1997.

[3] 이희락, "게임 사이트 구축을 위한 컴포넌트 설계" 공주대학교 석사학위 논문, 2003.

[4] Colin Atkinson. Meta-Modeling for Distributed Object Environment, 1997.

[5] 한국 더불린 코어 메타데이터, "메타데이터(matadata)란 무엇인가?", at URL : http://www.dublincore.or.kr/faq.htm, 2001.

[6] UKOLN Metadata Group. A Review of Metadata: A Survey of Current Resource Description Formats, 1998.

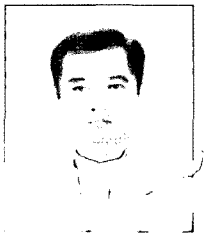
[7] 플라스틱 소프트웨어, "소프트웨어 모델링도구선택", at URL : http://agora.plasticsoftware.com/UMLKoea/View.aspx?brd=umlk_storage&pn=0&n=56000

[8] http://www.plasticsoftware.com/support/resource/PlasticComparision.pdf

[9] 볼랜드 자바팀, JbuilderStudyNet 공저, 최고의 자바개발 솔루션 JBuilder7, 가남사, 2002.

[10] http://www.borland.co.kr

[11] 국가지식정보통합검색시스템, http://www.knowledge.go.kr



김 치 수

e-mail : cskim@kongju.ac.kr

1984년 중앙대학교 컴퓨터공학과(학사)

1986년 중앙대학교 컴퓨터공학과(석사)

1990년 중앙대학교 컴퓨터공학과(박사)

1990년~1992년 공주교육대학교 전임강사

1992년~현재 공주대학교 컴퓨터공학부 교수

관심분야 : 소프트웨어 공학, 개발 방법론