응용프로그램 코드 생성기의 설계 및 구현

김영태*, 김치수*, 임재현**
*공주대학교 컴퓨터공학과
**공주대학교 컴퓨터 멀티미디어공학과
e-mail : zerot@kongju.ac.kr

Design and Implementation of Application Code Generator

Young-Tae Kim*, Chi-Su Kim*
*Dept of Computer Engineering, Kong-Ju National University
**Dept of Computer Multimedia Engineering, Kong-Ju National University

요 약

소프트웨어 개발에서 시스템 설계의 빈번한 변화와 그에 따른 프로그래밍의 어려움으로 인해 항상 시스템 설계와 구현 사이에 불일치하는 부분이 생기게 된다.

본 논문에서는 설계 모델을 영속적인 메타데이터로 저장하여 관리하며 추후에 설계의 변경이나 수정이 필요한 경우나 같은 영역의 시스템 개발 시에 적용할 수 있으며, 설계 모델로부터 코드를 생성하여 주는 도구의 개발을 통하여 설계와 구현사이의 불일치를 줄이고 소프트웨어 개발의 생산성과 유연성을 증대시킬 수 있는 방법을 제안하고자 한다.

1. 서론

현재 비즈니스 애플리케이션의 환경은 복잡하고, 다양하며, 빠르게 변화하는 현상을 보이고 있다. 이러한 현상에 능동적으로 대처하기 위해 소프트웨어 부품들의 재사용을 기반으로 하는 CBD나 객체지향 설계에서 반복되는 구조를 사용하는 디자인 패턴과 같은 객체기술을 적극적으로 도입해서 소프트웨어를 개발하고 있다[1]. 이는 소프트웨어 개발의 생산성 향상과 재사용을 추구하기 위한 방법이고 앞으로도 이 부분에 대한 노력은 계속될 것이다.

그러나 객체기술이 소프트웨어 개발 생산성과 품질향상을 위한 효과적인 개발 기법이기는 하지만 실제로 단기간 내에 고객의 요구에 따른, 혹은 비즈니스 도메인의 변화에 따른 애플리케이션의 개발이나 시스템 업그레이드 작업들은 매우 어렵고 때로 위태로운 작업이 되기도 하다.

또한 소프트웨어 개발에 있어 프로그래밍의 어려움과 시스템 요구사항의 빈번한 변화로 인한 시스템 설계와 구현사이의 문제점은 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.

본 논문에서는 설계와 구현사이의 문제점을 해결하기 위하여 시스템 설계로서의 비즈니스 객체 모델을 메타모델로 다루어 영속적인 메타데이터로 저장하고 관리하며 저장된 메타모델로부터 구현코드를 생성시켜주는 도구를 제시하여 환경의 변화나 요구사항에 맞도록 애플리케이션을 다시 작성하거나, 같은 영역의 시스템 개발 시에 소프트웨어 개발의 생산성 향상, 개발 시간의 단축, 그리고 좀 더 쉽고 유연하게 소프트웨어를 개발할 수 있는 개발 방법을 제시하고자 한다.

2. 관련연구

2.1 메타 모델링

메타모델링의 중요성에도 불구하고, 메타모델은 부족한 면을 많이 가지고 있다[2]. 실례로 메타모델의 본질과 구조에 대한 정의가 거의 없다.

객체 모델링 기술을 표현하는 메타모델링의 의미는 모델의 모델을 만들기 위한 것이다. 모델을 만드는 항목은 상위 레벨 모델의 항목으로 나타나는 인스턴스 뿐만 아니라 템플릿의 특징을 가지고 있다.

메타모델링 기술은 많은 측면에서 중요한 역할을 한다.

① 메타모델링은 전체 소프트웨어 공학과 관련된 데이터의 레포지토리에 대한 개념적 스키마로 사용할 수 있다.
② 메타모델링은 케이스 툴과 같은 모델링 툴의 개념적 스키마로 사용할 수 있다.
③ 메타모델링은 객체분석과 설계와 같은 모델링 언어를 정의하는데 사용할 수 있다.
④ 메타모델링은 서로 다른 모델링 언어에서 개념 사이의 관계를 이해하는데 도움을 주는 툴로서 사용할 수 있다.

본 논문에서 메타 모델링은 다양한 시스템 설계 모델을 정의하고 저장하는데 전반적인 개념이 사용된다.

2.2 메타 데이터

메타데이터에 대한 가장 단순하면서도 유용한 정의는 “데이터에 대한 구조화된 데이터”이다. 메타데이터란 어떠한 객체나 자원에 대한 서술 정보로서 메타데이터라는 말 자체는 상대적으로 새로운 것이지만 그 바탕이 되는 개념은 수집된 정보들을 조직화하기 시작했던 것만큼 오래 되었다고 할 수 있다[3]. DESIRE project에서는 메타데이터를 “객체의 존재와 특성을 보다 더 확실히 알고자 하는 잠재적인 사용자들로부터 객체를 해방시키는 것과 관련된 데이터”라고 기술하고 있다[4].

메타데이터의 목적은 데이터에 대한 실제 정보를 표현하는 것과 데이터 과정을 도와주는 것이다. 예를 들어 Java Class의 Attribute 데이터에 대한 메타데이터는 “attribute name”으로 불린다. 이 메타데이터는 주로 attribute에 대한 정보를 제공하는데 사용한다.

메타데이터는 실제 데이터를 표현하기에 맞도록 완전하고 정확해야 하며, 개발자들이나 시스템에 의해 사용되어지기 쉽고 융통성이 있어야한다

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

3. 설계와 구현 사이의 문제 해결 방안

본 논문에서는 객체설계를 일관된 객체구현으로 활용하기 위해 다음과 같은 방법을 제안한다.

① 같은 영역에 있는 비즈니스 애플리케이션을 공통의 영속적인 핵심 비즈니스 로직과 화면표시 로직으로 나누어 인식한다.
② 비즈니스 애플리케이션의 시스템 설계를 메타모델로 추출해서 비즈니스 로직은 클래스 메타모델로, 화면표시 로직은 GUI 메타모델로 다룬 후 영속적인 메타데이터로 메타데이터베이스에 저장한다.
③ 시스템 설계의 메타데이터를 편집할 수 있게 함으로써 비즈니스 애플리케이션의 커스터마이징에 초점을 맞추고 애플리케이션 커스터마이제이션을 데이터 입력 윈도우의 GUI 설계와 시스템 모델링과 연결한다.

시스템의 변화는 간단히 시스템의 메타데이터를 편집함으로써 새로운 코드가 생성된다. 그 결과 시스템 분석/설계의 산출물을 직접적인 구현으로 사용함으로써 시스템 분석/설계와 구현 사이의 문제점을 해결하는데 사용한다.

같은 도메인의 애플리케이션이 핵심 비즈니스 로직을 공유할 수 있기 때문에 시스템 모델과 시스템 외양을 포함하는 시스템 설계의 핵심이 공유될 수 있다. 같은 설계를 공유한다는 것은 모든 시스템이 같다는 것이 아니라, 단지 같은 설계패턴처럼 소프트웨어 구조를 공유한다는 것이다.

핵심 비즈니스 로직을 사용하는 장점은 거의 모든 애플리케이션이 필요로 하는 공통 솔루션을 제공해서 시스템 개발의 어렵고 과중한 업무를 현저히 감소시킨다. 또한 도메인 숙련가에 의해 제공되는 핵심 솔루션은 설계의 정확성과 시스템의 품질을 보증할 수 있다.

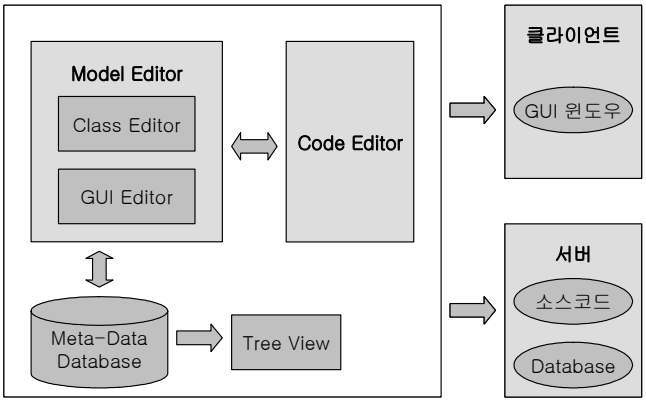
일반적으로 시스템 설계는 항상 UML 클래스 다이어그램이나 ER 다이어그램 같은 다이어그램 모델링으로 나타낸다. 이 모델링 다이어그램은 시스템에서 사용하는 비즈니스 클래스, 클래스에서 사용되는 속성과 메소드, 클래스들 사이의 관계 같은 것을 표현해 준다. 시스템을 구현하기 위해서 프로그래머는 다이어그램의 시스템 설계를 손수 컴퓨터 언어로 변환해야 한다.

본 논문에서 영구적인 메타데이터로 시스템 설계를 저장하려는 장점은 기존의 다이어그램이 단지 시스템 모델링을 표현할 수 있는 산출물인데 반해 메타데이터는 시스템 GUI 설계 정보, 비즈니스 로직 등의 정보들을 보다 상세하게 포함 할 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 것이다.

4. 시스템 설계 및 구현

어플리케이션의 소스코드와 데이터베이스 테이블의 자동 생성을 통한 설계와 구현의 연결은 시스템의 전형적인 객체 모델링을 생성할 수 있어야 한다. 그리고, 생성된 객체 모델링을 기반으로 하여 시스템의 GUI를 생성하고, 시스템 설계를 영구적인 메타데이터로 저장하여 수정할 수 있게 하고, GUI 화면을 수정할 수 있도록 하는 기능을 제공하여 커스터마이징이 가능하도록 하여야 한다.

다음 [그림 1]은 본 논문에서 제안하는 시스템의 구성도이다.

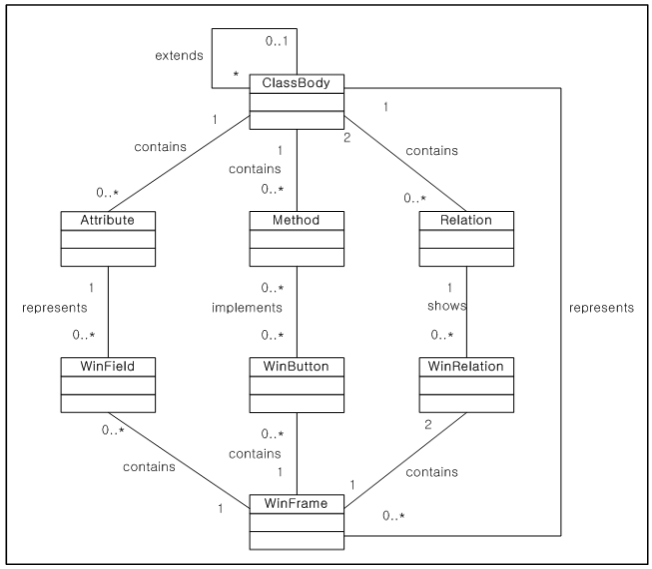


[그림 1] 시스템의 전체 구성도

4.1 Meta-Model 설계

시스템을 위해 시스템 데이터 모델을 기술하는데 사용되는 부분과 객체 모델에 기초를 둔 시스템을 위한 GUI 설계를 기술하는데 사용되는 두 부분으로 설계하되 독립적이지 않고 통합된 형태가 되도록 설계하여 시스템의 설계 모델과 GUI 모델이 상호 연동되어 동작하도록 함으로써 개별적으로 독립된 작업을 수행하면서 객체 모델과 시스템 GUI 디자인에 보다 많은 융통성을 부여하도록 한다.

다음 [그림 2]는 본 논문에서 제시하는 시스템의 Meta-Model로써 시스템 데이터 모델을 기술하는데 사용되는 4개의 클래스(ClassBody, Attribute, Method, Relation)와 GUI설계를 위해 사용되는 4개의 클래스(WinFrame, WinField, WinButton, WinRelation)로 구성되어 있다.



[그림 2] 시스템의 meta-model

4.2 Database 설계

시스템 내에서 메타 데이터로 사용하게 될 시스템 설계 데이터들을 저장하여 관리하기 위한 Database 설계의 장점은

① 메타데이터를 저장하는데 전체 8개의 테이블(ClassBody, Attribute, Method, Relation, WinFrame, WinField, WinButton, WinRelation)만 필요하다. 그 결과 데이터베이스 구조가 간단하고 분명하다.

② 각각의 테이블의 기본 키와 외래 키는 사용자가 요구하는 데이터를 빠르고 효율적으로 처리할 수 있게 해 준다.

③ 데이터베이스 설계는 완벽하게 메타모델 설계와 대등하다. 따라서 시스템의 구현과 수정을 쉽게 한다.

4.3 코드 생성 설계

메타데이터는 소스코드로 직접 매핑된다. ClassBody 객체는 서버 측 자바빈즈 클래스로 매핑된다. Attribute 객체는 자바빈즈 클래스의 속성으로 getter/setter 메소드와 함께 매핑된다. Method 객체는 자바빈즈 클래스의 메소드로 매핑된다. 자바빈즈 클래스에서 ContainsOne 타입을 가지고 있는 Relation 객체는 Relation 클래스에서 관련된 클래스에 의해 정의된 속성으로 관련된 객체에 매핑된다. 자바빈즈 클래스에서 ContainsMany 타입을 가지고 있는 Relation 객체는 Vector type 객체 속성으로

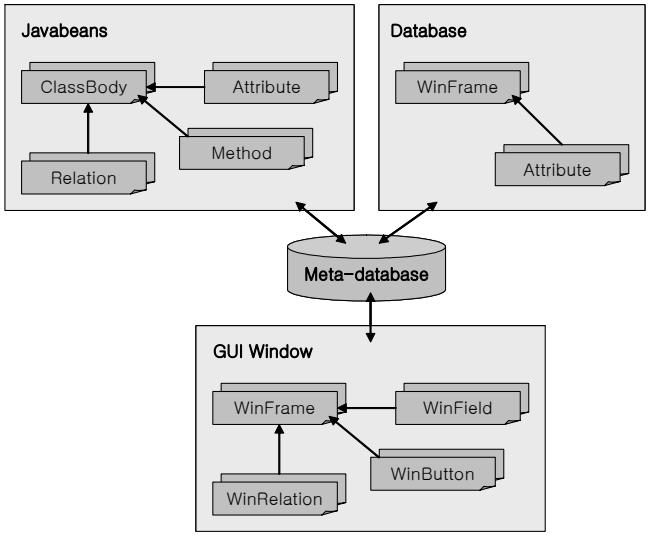
매핑된다.

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.

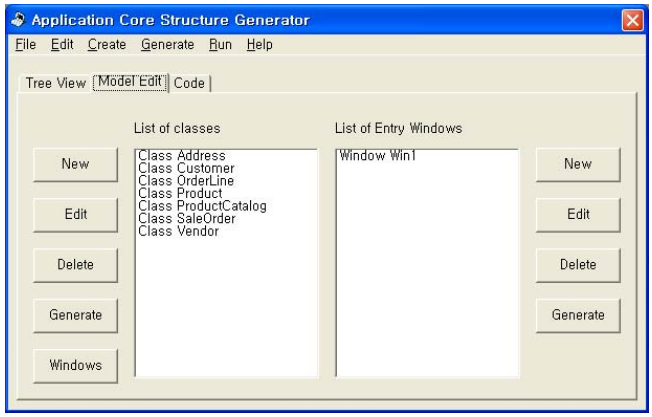
다음 [그림 3]은 코드 생성을 위한 설계도이다.



[그림 3] 코드 생성 설계도

4.3 시스템 구현

Tree View 부분은 시스템 모델과 관련된 정보를 사용자가 빠르게 찾을 수 있도록 하는 역할을 하며, Meta-Data Database에는 시스템 설계 정보가 메타데이터로 저장되어 추후에 설계 패턴으로 사용할 수 있는 정보를 제공하게 된다. Code Editor 부분은 모델설계를 통하여 생성된 코드를 편집하고 컴파일하는 기능을 수행한다. Model Editor 부분은 시스템의 핵심적인 부분으로써 GUI 설계를 담당하는 GUI Editor와 객체 모델과 관련된 처리를 담당하는 Class Editor 부분으로 구성된다. Class Editor 부분에서 시스템 설계 정보를 편집하여 설계를 변경할 수 있으며 변경된 설계 정보에 따른 코드가 생성되도록 한다. 다음 [그림 4]는 본 논문에서 제시하는 도구의 메인 화면으로 Model Editor 화면을 보여주고 있다.



[그림 4] 시스템의 메인화면

5. 결론 및 향후 연구과제

본 논문에서는 설계와 구현사이에서 발생하는 불일치를 줄이고 소프트웨어 개발의 생산성 향상 및 쉽고 유연한 소프트웨어 개발 방법의 제시를 위하여 설계 모델의 정의 및 수정이 가능하며, 설계 모델을 메타데이터로 저장하여 추후에 같은 영역의 소프트웨어 개발 시에 활용하고, 설계된 모델로부터 코드 생성이 가능한 통합 개발 환경을 제공하는 도구를 제시하였다. 향후 연구과제로는 본 논문에서 제시한 도구의 프로토타입을 기반으로 명확한 도구의 구현이 이루어져야 하며, 사용자 정의 패턴에 대한 효율성 검증의 연구도 이루어져야 할 것이다.

참고문헌

[1] ShallowayA., James R.T. Design Patterns Explained. Addison Wesley, 2002.

[2] Colin Atkinson, "Meta-Modeling for Distributed Object Environment", 1997

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

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