제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003. 5)

응용프로그램 핵심 구조 생성기의 설계에 관한 연구

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

A Study on the Design of Application Core Structure Generator

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

요 약

소프트웨어 개발은 보통 요구사항, 분석, 설계, 구현, 그리고 테스팅의 5단계의 과정으로 이루어진다. 시스템 설계의 빈번한 변화와 그에 따른 프로그래밍의 어려움으로 인해 항상 시스템 설계와 구현 사이에 불일치하는 부분이 생기게 된다.

본 논문에서는 비즈니스 로직을 인식하여 응용프로그램의 핵심적인 골격을 생성시켜주며, 설계 모델과 구현 코드를 상호 연동시켜 동적인 상호 변환을 통하여 설계 모델이 변경되거나 수정되면 그에 맞게 구현 코드가 수정되도록 하여 설계와 구현사이의 불일치를 줄이고 소프트웨어 개발의 생산성과 유연성을 증대시킬 수 있는 도구를 설계하고자 한다.

1. 서론

인터넷이 발달하면서 많은 무선 및 유선 분산 응용 소프트웨어가 개발되고 있고 앞으로도 계속될 것이다. 이러한 많은 개발들은 CBD 기반으로 이루어지고 있다. 이는 소프트웨어 개발의 생산성 향상과 재사용을 추구하기 위한 방법이고 앞으로도 이 부분에 대한 노력은 계속될 것이다.

현재 소프트웨어 개발을 위해 많은 GUI 기반의 소프트웨어 개발도구가 있고, 분석/설계를 위한 많은 UML 기반의 분석/설계 도구가 개발되고 사용되고 있다. 이 도구들 역시 마찬가지로 소프트웨어 개발의 생산성 향상과 개발 시간 단축을 위한 것들이다. 또한 앞으로 소프트웨어 개발을 좀 더 쉽고 유연성 있게 하기 위한 노력도 계속될 것이다[1].

그러나 분석/설계 도구를 사용하여 시스템을 설계하고 생성된 설계 모델을 가지고 코드 생성 기능을 이용하여 구현에 도움을 주고는 있지만 설계 모델이 변경되거나 수정되면 변경한 설계 모델이 구현에 바로 적용되는 것이 아니라 분석/설계 도구의 코드 생성 기능을 이용하여 새로운 구현 코드를 생성하여 적용하거나 개발자가 수작업을 통해서 일일이 구현 코드를 수정해야한다. 특히, 빈번한 설계 모델의 수정이 있는 경우 프로그래밍의 어려움은 더욱 커지게 된다.

본 논문에서는 설계와 구현사이에서 발생하는 불일치를 줄이고 같은 영역의 시스템들이 기본적인 설계 모델을 공유함으로 해서 소프트웨어 개발의 생산성 향상, 소프트웨어 개발시간의 단축, 그리고 좀 더 쉽고 유연한 소프트웨어 개발 방법의 제시를 목적으로 한다.

2. 관련연구

2.1 Object Modeling

Object Modeling은 설계 단계로서 요구사항과 분석 단계로부터 입력을 받아 시스템의 개념적이고 논리적인 설계 모델을 생성하는 객체지향 설계의 기본이다[2]. Object Modeling은 객체지향 시스템을 설계하고 소프트웨어 프로세스의 의사소통 언어를 제공하기 위해 사용되는 개념들을 정의하는 것이다.

성공적인 Object Modeling을 하기 위한 두가지 원칙은 다음과 같다.

- 시스템에 의해 요구되는 모든 객체들은 완전하고

정확하게 표현될 수 있도록 확실히 만들어야한다.
- 물리적 어플리케이션을 적용하기 위한 청사진처럼 시스템 개발자들에 의해 사용되어지기 충분하도록 세분화 되어져야 한다.

본 논문에서는 Object Modeling과 UML에 대한 연구를 통하여 객체지향 분석/설계에서 객체를 추출하고 표현하는 방법, Attribute를 추출하고 표현하는 방법, Operation을 추출하고 표현하는 방법 등에 대한 연구를 함으로써 UML을 이용하여 시스템의 Meta-Model을 설계하였다.

2.2 메타 데이터

메타 데이터는 데이터에 관한 데이터로써 실제 데이터나 작업에 대한 속성이나 내용을 기술한 것이다. DESIRE project에서는 메타데이터를 "객체의 존재와 특성을 보다 더 확실히 알고자 하는 잠재적인 사용자들로부터 객체를 해방시키는 것과 관련된 데이터"라고 기술하고 있다[3].

메타 데이터의 목적은 데이터의 실제 정보를 나타내며 데이터 과정을 도와준다. 메타 데이터는 file format, database table format과 같은 다양한 format들로 사용될 수 있으며, 메타 데이터가 실제 데이터를 표현하기에 완전하고 정확해야 하고, 또한 그들의 목적에 맞는 개발자들이나 시스템들에 의해 사용되기에 쉽고 융통성이 있어야 한다.

본 논문에서는 시스템 설계 정보를 영구적인 데이터로 저장하기 위해서 메타 데이터를 사용한다.

2.3 설계 패턴

설계 패턴은 시스템 설계시에 자주 발생하는 문제들에 대한 재사용 가능한 해결책이라 할 수 있다[4]. 경험 많은 소프트웨어 엔지니어의 해법들을 정리해서 이름을 부여하고 추상성을 주어 패턴화 시키면 미래의 비슷한 상황에서 다시 적용될 수 있게 되므로 시스템 설계시에 설계 재사용성을 제공할 수 있게 된다.

설계 패턴의 기본적인 특징은 다음과 같다[4][5].

1. 패턴은 설계시 일반적인 설계 방법을 제시하므로 설계자들에게 분석 및 도큐먼트를 쉽게 한다.
2. 패턴은 클래스와 객체 레벨의 상위에 정의되기 때문에 시스템의 복잡성을 줄인다.
3. 패턴은 시스템 개발시의 경험을 재사용할 수 있도록 한다. 이것은 설계도로써 이용될 수 있는 작은 구조도로 간주될 수 있다.
4. 패턴은 클래스 라이브러리에 대한 이해를 도움으로써, 초보 설계자들이 전문가처럼 수행할 수 있도록 도와준다.
5. 패턴은 설계의 재구성시의 비용을 줄인다.

본 논문에서는 설계 패턴에 대한 연구를 통하여 기존의 시스템 설계를 공유하는 것뿐만 아니라 설계 모델이나 코드로부터 사용자 자신의 패턴을 직접 생성하여 다른 어디에서도 사용할 수 있는 코드 템플릿을 작성할 수 있게 하고자 한다. 이렇게 만들어진 코드 템플릿은 어떤 패턴에도 포함 가능하도록 하여 같은 코드를 반복하여 사용하게 되는 경우 유용하게 사용함으로써 개발시간을 단축하는데 도움을 줄 수 있으며 팀 프로젝트를 하는 경우에 작성해 놓은 코드 템플릿을 사용함으로써 팀원간의 구현의 차이가 나타나는 것을 방지할 수 있도록 한다.

3. 시스템 요구사항

본 논문에서 제안하는 시스템은 다음과 같은 요구사항을 만족해야한다.

1) 같은 영역의 비즈니스 어플리케이션들이 비즈니스 로직과 프리젠테이션 로직의 일반적이고 안정적인 핵심 골격을 공유해야 한다. 즉, 기본 설계를 공유할 수 있어야 한다.

특별한 영역의 어플리케이션들을 살펴보면 다른 영역의 어플리케이션과는 다른 공통된 부분이 있음을 발견할 수 있다. 이런 공통된 부분들은 반드시 그 영역의 시스템들이 제공해야 하는 기능들을 담고 있음은 당연하다. 같은 영역의 어플리케이션들이 공통된 비즈니스 로직을 공유할 때 시스템 모델링과 시스템 프레젠테이션을 포함하는 시스템 설계의 핵심적인 부분을 공유할 수 있어야 한다. 시스템 설계를 공유하는 것은 시스템 전체가 같은 것이 아니라 설계 패턴과 같은 소프트웨어 구조를 공유하게 되는 것이다. 공통된 비즈니스 로직을 공유하게 되면 특정한 영역의 시스템들이 필요로 하는 공통된 솔루션을 제공할 수 있게 되어 시스템 개발자의 작업 부담감과 어려움을 경감시킬 수 있으며, 특정 영역의 전문가들에 의해 제공되는 핵심 솔루션을 사용하게 되면 설계의 정확함과 시스템의 품질을 보증 할 수 있게 된다.

2) 시스템 설계를 영구적인 데이터로 저장할 수 있어야 한다.

일반적으로 시스템의 설계는 다이어그램을 통해서 표현되어지는데 이 경우 시스템 설계 정보를 메타 데이터로 다루는 것이 가능하고, 메타 데이터를 데이터베이스에 영구적인 데이터로 저장할 수 있다.

영구적인 메타 데이터로 시스템 설계를 하는 것이 보다 더 실용적이고, 메타 데이터는 코드를 생성하고, 시스템을 형성하는 다양한 과정에 사용되어 질 수 있다. 다이어그램들은 시스템 모델링에만 사용될 수 있지만 메타 데이터는 시스템 GUI 설계 정보, 시스템 보안 정보, 비즈니스 로직 프로세스와 같은 다양한 정보를 담을 수 있다. 또한 시스템 설계의 메타 데이터를 편집함으로써 시스템 설계를 바꾸는 것이 가능하게 된다.

3) 기본 골격을 공유하면서 시스템 디자인과 프레젠테이션의 커스터마이징이 가능해야 한다.

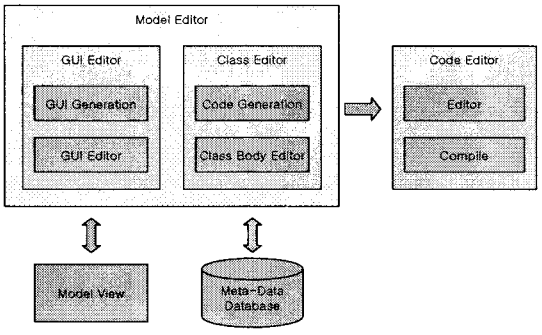
어플리케이션들이 항상 같을 수는 없다. 어플리케이션 개발 시 특정 영역의 핵심 로직을 공유하여 시스템을 설계한다고 하여도 개개의 어플리케이션들은 모두 고객의 요구에 맞게 작성되어 질 수 밖에 없다. 시스템 커스터마이징은 설계, 프레젠테이션, 점진적인 변화, 예기치 않은 변화 등에 대하여 다양한 단계들로 존재하게 된다. 설계의 커스터마이징은 시스템의 모델링과 GUI 설계를 변화시킴으로써 시스템을 변형시키는 것을 허가하는 것이며, 시스템을 위해 생산되는 새로운 코드의 결과일 수 있다. 프레젠테이션을 변형시키는 것은 시스템의 기능에는 영향을 주지 않는다. 점진적인 변화는 시스템이 어떤 기능이나 프로세스를 업그레이드 하거나 재정의 함으로써 변화되는 것이며, 예기치 않은 변화는 시스템의 일시적인 정지처럼 비상사태의 발생에 반응하는데 사용하게 된다.

본 논문에서의 커스터마이징은 설계와 프레젠테이션에 대해서 적용하게 되는데 모델링에 대한 커스터마이징은 시스템의 요구 조건에 따라서 변형될 수 있는 클래스들과 클래스의 속성, 메서드, 관계를 포함하는 시스템의 객체 모델을 의미하며, 프레젠테이션에 대한 커스터마이징은 GUI 화면의 구조가 변화될 수 있음을 포함하는 시스템의 인터페이스를 의미한다.

4. 시스템 설계

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 메타 데이터로 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. Model View 부분은 시스템 모델과 관련된 정보를 사용자가 빠르게 찾을 수 있도록 하는 역할을 하며, Meta-Data Database에는 시스템 설계 정보가 메타 데이터로 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.



[그림 1] 시스템의 전체 구조도

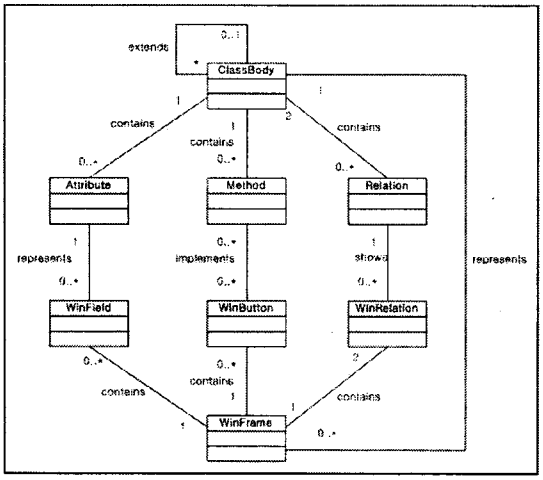
4.1 Meta-Model 설계

David Hay, Martin Fowler, Sanfrancisco에 의해 소개된 모델과 같은 일반적인 설계 모델들을 포함하고, GUI 설계들을 처리할 수 있으며, 단순 명료하고, 확장이 용이한 Meta-Model을 설계한다.

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 4개

의 클래스(WinFrame, WinField, WinButton, WinRelation)로 구성되어 있다.



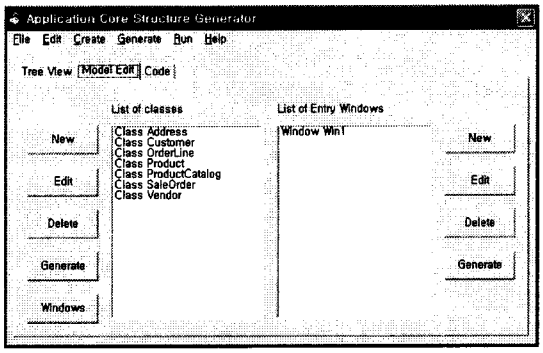
[그림 2] 시스템의 meta-model

4.2 Database 설계

시스템 내에서 메타 데이터로 사용하게 될 시스템 설계 데이터들을 저장하여 관리하기 위한 Database를 설계한다. 설계 패턴을 위해 사용될 설계 모델이나 코드 템플릿에 대한 정보를 저장하여 관리함으로써 어느 곳에나 사용될 수 있는 안전한 데이터의 역할을 할 수 있도록 한다. 데이터베이스는 전부 8개의 테이블(ClassBody, Attribute, Method, Relation, WinFrame, WinField, WinButton, WinRelation)이 필요하며, 데이터베이스 구조는 단순하고 명료하게 설계한다. 각각의 테이블에 있는 primary key와 foreign key를 이용하여 사용자들이 빠르고 효율적으로 데이터를 액세스하게 한다.

4.3 시스템 구조 및 인터페이스 설계

시스템 설계 모델의 메타 데이터를 저장, 관리하며, 코드 생성이 가능하도록 시스템의 구조를 설계하고 GUI 설계를 편집하고 수정하는데 용이한 인터페이스를 설계한다. 객체 모델과 관련된 처리를 담당하는 Class Edit 부분과 GUI 구성과 관련된 처리를 담당하는 GUI Edit 부분으로 구성된다. 객체 모델과 GUI 설계가 상호 호환되도록 하여 설계와 구현을 이어줄 수 있는 구조를 갖도록 설계한다. 인터페이스는 시스템 설계의 메타 데이터를 사용자가 편집하는 방식으로 제공하며 시스템의 GUI 화면이 어떻게 보여질 지에 대한 감각을 제공하도록 한다. 다음 [그림 3]은 본 논문에서 제시하는 도구의 메인 화면이다.



[그림 3] 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 설계 패턴으로 활용하고, 설계된 모델로부터 코드 생성이 가능한 통합 개발 환경을 제공하는 도구를 설계하였다. 향후 연구과제로는 본 논문에서 제시한 도구의 프로토타입을 기반으로 명확한 도구의 구현이 이루어져야 하며, 사용자 정의 패턴에 대한 효율성 검증에 대한 연구도 이루어져야 할 것이다.

참고문헌

[1] J. P. Gray, A. Liu, and L. Scott. "Issues in Software Engineering Tool Construction", Information and Software Technology, 2000

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

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

[4] Erich Gamma, Richard Helm, Ralph Johnson, John Vissides 공저, 김정아 역, "GoF의 디자인 패턴", 2002

[5] E. Gamma, R. Helm, R. Johnson, J. Vissides, "Design Patterns: Abstraction and reuse in object-oriented design," Proceedings of ECCOOP'93, 993