2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

패턴적용을 통한 기능분할 GUI 컴포넌트 구조 모델

김운용○ 최영근
광운대학교 컴퓨터과학과
{wykim, ygchoi}@cs.kwangwoon.ac.kr

The Architectural Model of Function Dividing GUI Component by Pattern-Adaption

Woon-Yong Kim○ Young-Keun Choi
Dept. of Computer Science, Kwangwoon University

요 약

통합된 개발 GUI 컴포넌트의 사용은 어플리케이션을 거대하게 만든다. 시스템 개발 시 개발된 GUI 컴포넌트의 적용은 분야에 따라 사용용도 및 특징들이 매우 다양하다. 그러나 GUI 컴포넌트는 이러한 사용자의 요구에 충분히 만족하여야하는 특징을 가지고 있다. 이에 본 논문에서는 GUI 컴포넌트의 이러한 두 가지 특징을 만족하면서 작고, 최적화되며, 재사용 가능한 컴포넌트 구조 모델을 제시하고자한다. 제시된 모델에서는 컴포넌트에 필요한 기능단위를 분할하여, 시스템 개발의 필요시 사용용도와 특징을 고려하여 다양한 형태로 조합 가능하게 함으로써 최적화된 시스템을 구축할 수 있도록 한다. 또한 컴포넌트 구조 모델을 객체 지향 디자인 패턴에 적용함으로써 일관되고 효율적인 구조 모델을 제시할 수 있다.

1. 서론

소프트웨어 패러다임 변화의 주된 요인은 소프트웨어 재사용성 향상과 성능의 최적화 및 편리한 프로그램 개발에 있다. 현재 객체 지향 기술을 통한 소프트웨어 개발은 무엇보다도 실체에 대한 프로그램 적용이라 할 수 있다. 이러한 적용을 위해 가장 중요시되는 것이 객체 단위의 재사용이라 할 수 있다. 또한 이러한 객체 단위의 재사용에 대한 가장 일반적으로 사용 가능한 구성 형태를 패턴으로 분리하여 관리하고 있으며 시스템 분석과 설계시 이용되어진다. 대표적인 패턴으로 GUI 컴포넌트 개발시 이용되는 MVC 모델[7]이라 할수 있다. 본논문에서는 이러한 MVC 모델의 확장을 통한 기능 분할 컴포넌트 표현 방법을 제시하고자한다. 이러한 목적은 사용자의 요구에 맞는 시스템 개발시 사용용도와 특징들이 매우 다양화되어있으며, 이러한 사용자의 요구를 충족하는 컴포넌트 개발시 모든 사용용도와 특징들을 고려 하여 개발 되어야 함으로 프로그램이 거대해 진다. 그러나 컴포넌트를 사용하는 용도에 따라 부분적으로 이용되므로 불필요한 자원과 실행속도저하를 가져온다. 이러한 요소를 해결하기 위해 본 논문에서는 확장 MVC 모델을 제시하고 이 모델에 의한 컴포넌트를 사용용도와 특징을 고려하여 재구성 함으로써 불필요한 자원의 사용을 줄이고 최적화된 컴포넌트를 구성할 수 있도록 하고, 컴포넌트 구조모델을 디지인 패턴에 적용함으로써 일관되고 효율적인 모델 방법을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 소프트웨어 설계시 널리 이용되는 디자인 패턴과 GUI 컴포넌트 구성 모델 및 설계 방향을 제시하고 3장에서는 기능 분할 컴포넌트 구조 모델과 특징을 설명한다. 4장에서는 구성된 기능 분할 컴포넌트를 이용한 재구성 시스템를 소개하고, 대표적인 MVC 모델인 Swing 컴포넌트를 이용한 기능분할 컴포넌트 구현과 적용 방법을 제시한다. 마지막으로 5장에서는 결론 및 향후과제를 보인다.

2. 관련연구

이장에서는 기능분할과 컴포넌트 구성과 관련된 디자인 패턴과 MVC 모델에 대한 내용을 분석한다.

2.1 디자인 패턴

디자인 패턴은 디자인 경험을 표현하기 위한 새로운 매커니즘으로써 제안되었다. 이러한 디자인 패턴은 미래의 비슷한 상황에서 다시 적용될 수 있는 과거에 잘 정의된 설계에 대한 정보를 기록하는 것이다.[8] 또한 이러한 패턴들은 소프트웨어 시스템을 구성하는 빌딩블럭으로서 사용되어진다. 이러한 패턴은 크게 프레임 워크를 형성하는 아키텍쳐 패턴 과 시스템 단위를 형성하는 객체지향 디자인패턴 그리고 코딩 패턴으로 분리되어진다.[6]

2.2. 패턴의 정의 및 특징

디자인 경험을 표현하는 새로운 메커니즘인 패턴은 다음과 같이 정의되어진다.

> *"A Pattern is a cluster of cooperating objects linked by certain relationships and the roles which express a link between a context the design problem and its solution and becomes the module of a unitary architectural model[2]"*

이러한 패턴의 기본적인 특징은 다음과 같다.[5]

1. 패턴은 설계시 일반적인 설계 방법을 제시한다. 그러므로 이것은 설계자들에게 분석 및 도큐먼트를 쉽게 한다.
2. 패턴은 클래스와 객체 레벨의 상위에 정의 되기 때문에 시스템의 복잡성을 줄인다.
3. 패턴은 소프트웨어 개발시의 경험을 제사용할수 있도록한다. 이것은 설계도로써 이용될 수 있는 작은 구조도로 간주될 수 있다.
4. 패턴은 클래스 라이브러리에 대한 이해를 도움으로써, 초보 설계자들이 전문가처럼 수행할수 있도록 도와준다.
5. 패턴은 설계의 재구성시의 비용을 줄인다.

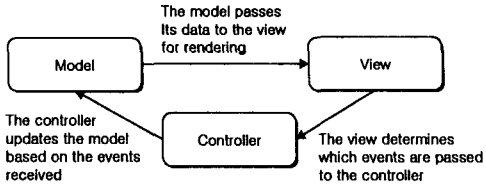
	Patterns		
	Creational	Structural	Behavioral
Names	Factory Method	Adapter	Chain of Responsibility
	Abstract Factory	Composite	Command
	Builder	Flyweight	Interpreter
	Prototype	Facade	Iterator
	Singleton	Bridge	Mediator
		Decorator	Observer
			State
			Strategy
			Template Method
			Visitor

(표 1) 디자인 패턴의 종류

패턴은 아키텍쳐 패턴과 객체지향 디자인 패턴 그리고 코딩 패턴으로 분리되어진다.[4] 먼저 아키텍쳐 패턴은 전체적인 시스템 구조를 위한 패턴으로 만들고자하는 어플리케이션에 의존한다. 소프트웨어 시스템의 이러한 아키텍쳐 패턴은 어플리케이션의 기본구조를 결정한다. 또한 객체지향 디자인 패턴은 단위 프로그램을 구성하기위한 패턴으로 Gamma에 의해 Creational, Sturctural, Behavioras(생성, 구조, 행위) 형태의 그룹으로 분리된다.[5] 종류는 (표 1)과 같다.

2.3 MVC 모델

MVC(Movel-View-Controller)는 객체지향 사용자 인터페이스의 분해을 위해 잘 알려진 방법으로 컴포넌트의 기반 설계 방식이다.[7] 이러한 MVC는 GUI컴포넌트를 세가지 요소로 분리하여 관리한다. 세가지요소인 모델, 뷰, 컨트롤러는 컴포넌트의 행위에 대한 각각의 역할을 수행한다. 이러한 모델을 표현하면 (그림 1)과 같이 표현된다. 그림에서 보여지듯이 이 MVC 모델은 각 요소들간의 책임할당과 작은 의존성을 가지고 구성되어진다.



(그림 1) MVC 모델 구조와 관계

MVC 모델은 하나의 모델에 여러개의 뷰와 컨트롤을 통하여 구성할 수 있다. 그러나 이러한 구성은 여러기능들을 포함함으로써 불필요하게 거대해 질 수 있는 단점을 가진다. 이러한 문제를 해결하기 위해 본 논문에서는 기능분할을 통해 컴포넌트를 분리할 수 있는 모델을 제시하고 다양한 분야와 특징에 접합한 최적의 컴포넌트를 구성하는 방법을 제시한다.

3. 기능분할 컴포넌트 구조 모델

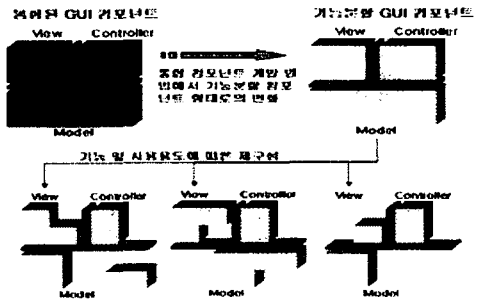
이 장에서는 기능 분할 컴포넌트 구성시의 특징과 구조 모델을 제시한다. 기능분할 컴포넌트 구조 모델은 기존의 MVC 모델에 Facade 패턴을 적용하여 모델과 컨트롤를 인터페이스를 통해 통합된 형태로 제시함으로써 사용용도와 특징에 의해 조합 가능한 View를 생성할 수 있는 방법을 제시할 수 있다.

3.1 기능 분할 컴포넌트의 특징

기능 분할 컴포넌트의 특징은 하나의 컴포넌트를 다양한 사용목적에 따라 코드의 수정없이 적용하고, 이때 목적에 알맞은 기능들로 재구성하는 메커니즘을 제공한다. 이렇게 함으로써 다음과 같은 비용을 절감할 수 있다.

1. 클래스 생성비용
2. 메모리 사용비용
3. 관리비용
4. 실행속도 비용

이러한 형태의 기능분할 모델형태는 (그림 2)와 같다.



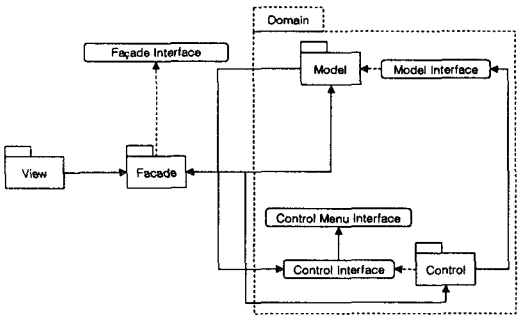
(그림 2) 기능 분할 컴포넌트 개념도

3.2 기능 분할 구조 모델

기능 분할 구조 모델은 이러한 기능단위를 효율적으로 분리하는 모델을 제시한다. 모델 구성의 기본원리는 MVC 모델의 구조와 이들간의 관계는 공통성질을 가지는 인터페이스로 연결

하고 이 인터페이스를 상속받은 각 MVC를 생성함으로써 서로 다른 특징을 가진 형태를 공통의 인터페이스에 의해 연결함으로써 인터페이스 조작을 통해 서로간의 자유스러운 통신을 제공한다. 또한 여러 View를 만들기의해 View의 추상 View인 Facade Package를 만들어 새롭게 구성될 View는 이 Facade Package만을 참조함으로써 복잡성을 간소화 시키는 역할을 담당하도록 구성될 수 있다. 이 절에서는 기능 분할과 관련된 MVC의 관계를 도식화하고 각 구성들간의 관계와 특징 및 구조모델 설계 방법를 제시한다.

3.2.1 기능 분할 컴포넌트의 구조 모델



(그림 3) 기능 분할 컴포넌트 구조 모델

(그림 3) 기능 분할 구조 모델은 각 패키지들간의 관계와 인터페이스관계를 나타낸다. 이 모델에서 MVC 모델은 VFMC(View-Facade-Model-Control) 모델로 확장된다. 또한 Facade, Model, Control은 서로간의 공통된 통신을 의해 정의된 각각의 Interface를 상속받아 작성되어진다. 이렇게 함으로써 일관되고 공통된 연결이 가능하게된다. 각 Model과 Control은 기능단위로 분할하여 각 패키지 단위로 구성되어진다. 또한 View는 Model과 Control 패키지에 대한 고려 없이 Facade 패키지의 정보를 활용하여 사용용도와 특징에 맞는 View들을 구성할 수 있다.

3.2.2 기능 분할 구조 모델 설계 방법

기능 분할 설계 방법은 인터페이스 설계에서부터 구성요소 분할, 클래스 생성, 사용용도 적용의 특징을 고려하여 4단계로 구성되어 진다.

1 단계: 컴포넌트 구성에 필요한 기능 분할을 통한 각 패키지의 인터페이스 설계

2 단계: 패키지 단위의 구성 요소 분할 설계

3 단계: 패키지 단위로 분할된 구성요소를 기반으로 해당 인터페이스를 상속받은 클래스 생성. 이 단계에서 패키지 안의 클래스들간의 관계를 설정하기 위해 필요한 패턴을 적용시킨다.

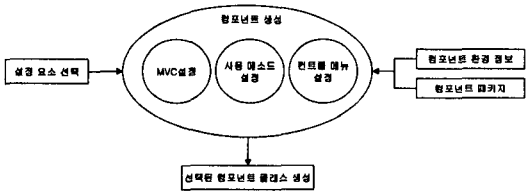
4 단계: Facade 패키지의 클래스들을 이용한 사용용도 및 특징에 맞는 다양한 뷰 생성 및 적용

각 기능 모델에 포함된 요소는 패키지 단위로 분리되어지며 패키지들은 공통 인터페이스를 통해 서로간의 통신이 이루어지므로 패키지단위의 의존성을 줄이고 독립된 형태로 구성되어질 수 있고, 새로운 기능의 추가 및 수정이 전체 구조의 변화없이 이루어 질 수 있다.

4. 기능 분할 컴포넌트 재 구성 시스템

기능 분할 컴포넌트 재구성 시스템은 사용용도와 필요성에 따라 다양한 형태의 컴포넌트를 구성하기 위한 시스템으로 기능 분할 구조 모델에 기반한 컴포넌트를 사용자의 요구에 기반한 다양한 형태로 구성할 수 있도록 한다. 기능 분할 구조 모델이 공통 인터페이스 기반 설계 구조를 가지므로써 이러한 인터페이스의 기능을 조합함으로써 다양한 형태를 구성할수 있으며, 자동화된 컴포넌트 생성을 가능하게 한다. 재구성 시스템 구성도는 (그림 4)과 같이 구성된다.

4.1 기능 분할 컴포넌트 재 구성 시스템 구성

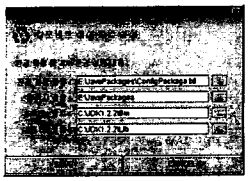


(그림 4) 기능 분할 컴포넌트 재구성 시스템 구성도

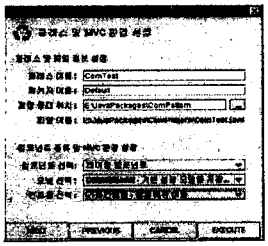
기능 분할 컴포넌트 재구성 시스템은 크게 입력 부분으로 설정요소 선택부분과 컴포넌트 패키지 부분이 있으며 이 내용을 기반으로 컴포넌트 생성 프로세스를 통해 필요한 설정요소에 기반한 컴포넌트 클래스를 생성하는 출력 부분으로 구성되어진다. 컴포넌트 생성시 MVC 모델의 선택과 프로그램에 사용될 메소드 및 컨트롤 메뉴 설정을 통해 필요한 요소를 선택하여 기능에 필요한 최적화된 컴포넌트를 구성할 수 있다.

4.2 기능 분할 컴포넌트의 구현 및 적용

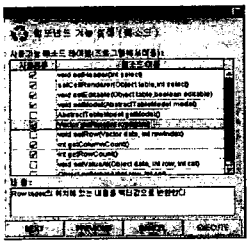
이러한 시스템을 구현 및 적용하기 위해 본 논문에서는 MVC 모델의 대표적인 특징을 가지고 있는 자바 Swing 컴포넌트를 대상으로 기능 분할 컴포넌트 모델을 구성하고 이 모델 기반으로 용도 및 필요성에 적합한 컴포넌트를 생성할 수 있는 시스템을 구현한다. (그림 4)의 시스템 구성도에서 보여지듯이 이러한 요소 선택과 컴포넌트 패키지 활용을 위한 시스템을 구성한다. 이러한 기능분할에 의해 생성된 기능 분할 컴포넌트 모델를 사용자의 요구에 따라 적절하게 결합함으로써 필요에 충족하는 컴포넌트를 재구성하기 위해 이용되어진다. (그림 6)의 환경 설정을 통해 기능 분할 모델로 구성된 클래스 패스를 설정하고, 필요한 컴포넌트를 구성하기 위한 사용자 요구로 MVC환경설정과 프로그램에서이용가능한 컴포넌트에 적용되는 메소드 정의 및 컨트롤 중에서 팝업 메뉴에 필요한 메뉴설정과 같은 사용자요구를 받아 요구에 적합한 컴포넌트 구성요소를 결정한다. 이러한 환경설정은 (그림 6~8)에서 보여준다.



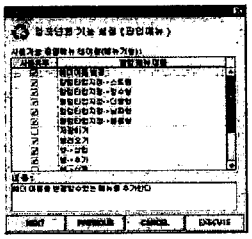
(그림 5) 모델 환경설정



(그림 6) 클래스 및 MVC환경

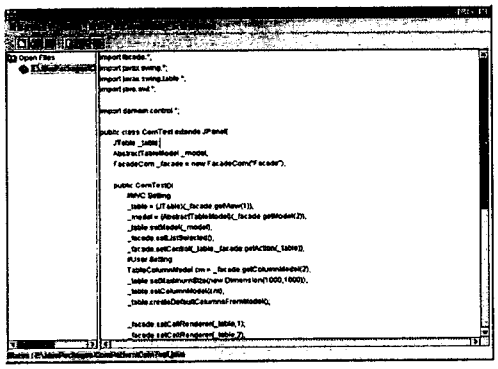


(그림 7) 컴포넌트 기능 설정



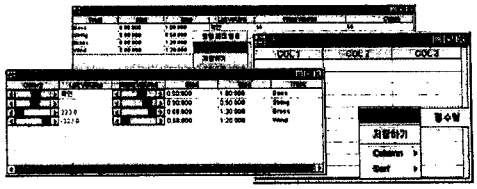
(그림 8)컴포넌트 메뉴설정

각 기능에 필요한 환경 및 메소드 메뉴 설정을 통해 이들 컴포넌트 생성 프로시저에 의해 기능에 적합한 클래스파일이 (그림 9)과 같이 생성되어진다. 이렇게 생성된 컴포넌트는 업무에 필요한 개발시 내부 컴포넌트로 활용되어진다.



(그림 9) 재구성 통합환경 및 생성 클래스 결과

(그림 10)은 기능 분할 형태로 구성된 컴포넌트에서 사용자의 요구에 따라 다양한 형태의 테이블이 구성되어진 결과를 보여준다. 이렇게 구성된 컴포넌트는 분야별로 적절하게 기능을 포함하고서 활용될 수 있으므로 불필요한 메모리의 낭비를 최소화 하며 수행속도 향상을 가져올 수 있다.



(그림 10) 요구기능에 따른 컴포넌트들

5. 결론 및 향후과제

본 논문에서는 MVC 모델기반 GUI컴포넌트를 요구단위 기능 분할을 통해 사용자의 요구에 따라 다양한 형태로 구성될 수 있는 확장 MVC모델인 VFMC방식을 제시하고 이러한 구성을 위한 디자인 패턴을 적용하였으며, 이렇게 구성된 확장 MVC모델을 이용하여 다양한 컴포넌트 활용을 위한 통합 컴포넌트 제구성 시스템을 제시하였다. 이러한 구성은 네트워크화된 컴퓨팅 분야를 통한 작은 컴포넌트 요구와 수행성능에 밀접하게 연관된 GUI부분에서의 기능요구에 부합된 작은 컴포넌트를 구성함으로써, 메모리의 효율성을 높이고, 불필요한 자원의 사용을 방지하는 효과를 가져옴으로써 보다 효율적인 컴포넌트를 구성할 수 있게할 수 있다. 또한 이러한 모듈기반 컴포넌트 구성은 GUI 뿐만 아니라 소프트웨어 개발의 다양한 분야에서 활용될 수 있을 것이다.

참고문헌

[1] Alexander, Christopher, S. Ishikawa, et al. A Pattern Language. New York: Oxford University Press, 1997.

[2] Amund Aarsten, Davide Brugali and Giuseppe Menga, "Designing Concurrent and Distributed Control Systems: an Approach Based on Design Patterns", Communications of the ACM, October 1996.

[3] Arnold, Ken, and J. Gosling. The Java Programming Language. Reading, Mass.: Addison-Wesly, 1998.

[4] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: elements of Reusable Object-Oriented Software, Addison-Wesley, 1995

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns: Abstraction and reuse in object-orientted designs," Proceedings of ECCOOP'93, pp. 406-431, 1993

[6] F. Buschmann and R. Menuier. "A System of Patterns", "Pattern Language of Program Design, Coplien and Schmidt, eds., Addison-Wesley, 1995.

[7] Krasner, G. E., and S. T. Pope. "A Cookbook of Using the Model-View-Controller User Interface Paradigm in Smalltalk-80.", Journal of Object-Oriented Programmkng I(3), 1998.

[8] K. Beck, J. O. Coplien, R. Crocker, L. Dominick, G. Meszaros, F. Paulisch, and J, Vlissides, "Industrial Experience with Design Patterns," Procedings of ICSE-18, 1996