관찰자/피관찰자 설계 패턴을 이용한 모바일 M/VC 응용 프레임워크☆

Mobile M/VC Application Framework Using Observer/Observable Design Pattern

음 두 헌*
Doohun Eum

요 약

최근, 휴대폰과 PDA 등의 무선기기 사용자가 급증하고 있다. 이에 따라 지리정보, 교통안내 시스템과 같은 모니터링 및 제어 응용이 무선기기에서 활발히 사용되고 있다. 본 논문에서는 모바일 응용 중, 모니터링 및 제어 응용인 M/VC(Model/View Controller) 응용의 신속한 작성을 지원하는 모바일 M/VC 응용 프레임워크를 소개한다. 모바일 M/VC 응용 프레임워크는 무선 통신 환경에서 클라이언트와 서버 객체의 상호작용을 자동 처리하기 위해 Java의 관찰자/피관찰자(Observer/Observable)를 확장한 모바일 관찰자/피관찰자 패턴과 관찰자/피관찰자 객체들의 조립 기능을 제공하는 Multiplexer와 Demultiplexer 클래스들을 지원한다. 개발자는 이 프레임워크를 이용하여 Observable과 MobileObserver 클래스들로부터 필요한 객체들을 생성한 후, 이들을 Multiplexer와 Demultiplexer 객체에 구성적으로(plug-and-play식으로) 상호 연결하여 응용을 생성한다. 즉, 개발자는 무선 환경을 고려하지 않고 모바일 M/VC 응용 프레임워크가 제공하는 Multiplexer나 Demultiplexer 클래스의 객체에 모바일 관찰자/피관찰자 객체들을 조립식으로 연결함으로써 피관찰자의 상태 변화가 관찰자에게 전달되고 관찰자를 통한 사용자의 입력이 피관찰자에게 전달되어 반영되는 모바일 모니터링 및 제어 응용을 신속하게 작성할 수 있다. 또한, 모바일 M/VC 응용 프레임워크는 무선 통신 환경하의 관찰자/피관찰자 객체들과 같은 컴포넌트 재사용성을 개선한다.

Abstract

Recently, the number of mobile phone and PDA users has been rapidly increased. Such monitoring and control applications as geographical and traffic information systems are being used widely with wireless devices. In this paper, we introduce the mobile M/VC application framework that supports the rapid constructions of mobile monitoring and control (M/VC) applications. The mobile M/VC application framework uses the mobile Observer/Observable pattern that extends the Java's Observer/Observable for automatic interactions of server and client objects in wireless environments. It also provides the Multiplexer and Demultiplexer classes that supports the assembly feature of Observer and Observable objects. To construct an application using the framework, developers just need to create necessary objects from the Observable and MobileObserver classes and inter-connect them structurally (like the plug-and-play style) through the Multiplexer and Demultiplexer objects. Then, the state change of Observable objects is notified to the connected Observer objects and user's input with Observer objects is propagated to Observable objects. These mechanism is the main process for monitoring and control applications. Therefore, the mobile M/VC application framework can improve the productivity of mobile applications and enhance the reusability of such components as Observer and Observable objects in wireless environments.

☞ Keyword : application generators, application frameworks, mobile applications, Observer/Observable pattern, MVC model.

1. 서 론

2004년 10월말 유·무선 통신 서비스 가입자 현황을 살펴보면, 무선기기 사용자의 점유율이 60% 이상이며 무선기기 가입자 수는 이미 3천 5백만 명을 넘어섰다. 앞으로는 무선기기 가입자 시장보다는 다수의 가입자를 기반으로, 무선

* 정 회 원 : 덕성여자대학교 컴퓨터공학부 교수
dheum@duksung.ac.kr
[2005/10/10 투고 - 2005/10/29 심사 - 2006/01/23 심사완료]
☆ 본 연구는 2005년도 덕성여자대학교 자연과학연구소의 연구비 지원에 의해 수행되었음.

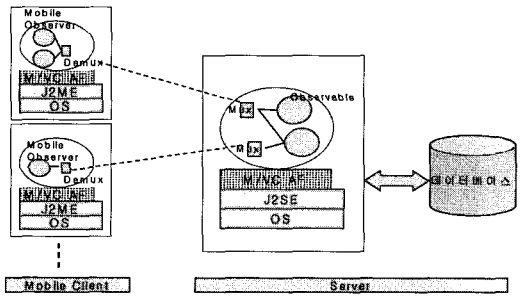
기기의 이동성과 용이성을 활용한 모니터링 및 제어 응용 보급에 주력하여야 한다[1]. 현재는 생산공정 시스템, 지리정보 시스템, 교통안내 시스템 등과 같은 모니터링 및 제어 응용이 무선 기기에서 활발히 사용되고 있다. 소비자가 직접 구매하는 장소인 소매점을 여러 개 관리하는 회사와 물품의 재고를 비축해 놓고 판매하는 도매점들로 구성된 생산공정 시스템의 경우, 회사는 도소매점들의 재고량 수준을 모니터링하고 공급자에 대해서는 가격을 기준으로 적절한 공급자를 결정한 후, 물품의 주문 및 재고를 제어해야 한다. 또한, 택시 회사와 같은 곳에서도 모니터링 및 제어 응용을 도입한 교통제어 시스템을 사용 할 수 있다. 회사는 택시들의 위치를 모니터링하고 한 지역에 택시들이 밀집되어 있으면 밀집되지 않은 곳으로 분산시켜야 한다. 댐 수문 제어 응용에서도 수문을 열고 닫아야 할 때를 모니터링 하여 수문의 열고 닫음을 제어할 수 있다. 이러한 모니터링 및 제어 응용에 모바일 관찰자/피관찰자 객체를 사용하면 객체들이 실시간으로 상호 작용하므로 정보의 변화를 시간과 장소의 제약 없이 신속하게 반영할 수 있다.

모바일 응용의 생산성 향상을 위해 무선 환경의 모델링에 적합하고 재사용성이 뛰어난 객체지향 기술이 많이 사용되고 있고, GUI(Graphical User Interface) 시스템이나 클라이언트/서버 시스템처럼 복잡한 기능을 갖는 객체들을 분류하는 방법론 중 하나인 MVC(Model, View, Controller) 패턴도 많이 사용되고 있다. MVC 패턴은 모바일 응용을 모델, 뷰, 컨트롤러의 3개 모듈들로 나누어 각 모듈에 프로그램의 특정한 역할을 할당한다[2]. 모델은 응용 프로그램의 내부 상태 정보를 표현하는 데이터를 유지·관리한다. 뷰는 출력 장치 상의 모델의 시각적 표현에 대한 작업을 담당한다. 하나의 모델에 여러 개의 뷰가 존재할 수 있다. 컨트롤러는 사용자의 입력을 유도하는 역할을 담당한다. 이러한 역할 분담을 전제로 구현된 응용 프로그램은 세 부분이 상호작용함으로써 수행된다. 여기서, 뷰와 컨트롤러는 일반적으로 GUI에 해당되는 요소들로서 결합되어 제공되므로 MVC 모델을 M/VC 모델로 표기할 수 있다[3]. 또한, 객체들 간의 상호작용을 자동 처리하기 위해 Java의 관찰자/피관찰자(Observer/Observable)[4]를 모바일 관찰자/피관찰자 패턴으로 확장하여 사용할 수 있다. 모바일 M/VC 응용 프레임워크는 모바일 M/VC 응용의 신속한 조립식 작성을 위해 객체들의 조립성을 지원하는 Multiplexer와 Demultiplexer 클래스들과 모바일 관찰자/피관찰자 패턴을 구현한 MobileObserver/Observable 클래스들을 지원함으로써 구성적 인터페이스(structural interface)를 지원한다. 구성적 인터페이스는 기존 객체지향의 절차적 인터페이스(procedural interface)에 비해 객체 간의 plug-and-play식의 조립을 통한 연동을 가능하게 한다.

모바일 응용의 구조에는 무선기기에 마이크로브라우저만 요구되는 얇은 클라이언트(thin client) 구조와 J2ME와 같은 프로그램 실행 환경이 요구되는 스마트 클라이언트(smart client) 구조의 2가지가 있다[5, 6]. 얇은 클라이언트 구조는 서버 측과의 무선 연결이 항상 유지되어야 하며 스마트 클라이언트 구조는 데이터 동기화를 위해 간헐적인 연결이면 충분하다. 모니터링 및 제어 응용인 모바일 M/VC 응용은 클라이언트와 서버가 항시 연결되어야 하며 클라이언트 측에 모바일 응용이 실행되어야 하므로 두 구조의 중간 형태의 구조를 필요로 한다.

그림 1은 모바일 M/VC 응용 프레임워크를 이용하여 생성한 모바일 응용의 구조이다. 모바일 클라이언트 측은 무선기기의 운영체제 상에 J2ME 플랫폼이 위치한다. J2ME 플랫폼은 사용되는 무선기기별로 Java 플랫폼을 정의하는 프로파일을 제공하여 모바일 응용의 개발과 실행을 가능하게 한다[6, 7]. J2ME 플랫폼 상에 모바일 M/VC 응용 프레임워크가 위치하며 모바일 M/VC 응용 프레임워크는 Multiplexer, De-

multiplexer, MobileObserver 등의 클래스들을 포함한다. MobileObserver와 Demultiplexer 클래스를 상속받는 클래스를 생성하여 모바일 응용의 클라이언트 측을 작성한다. 서버 측에는 운영체제 상에 J2SE 플랫폼이 위치하고 그 위에 모바일 M/VC 응용 프레임워크가 위치한다. 모바일 M/VC 응용 프레임워크의 Demultiplexer 클래스를 상속받는 클래스와 J2SE가 제공하는 Observer/Observable 클래스의 객체들을 생성하여 서버 응용이 작성된다.



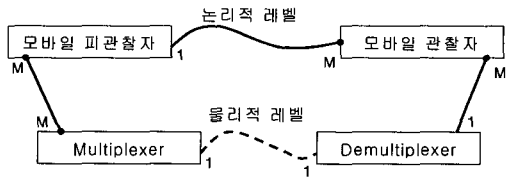
〈그림 1〉 모바일 M/VC 응용 프로그램의 구조

모바일 클라이언트 측의 MobileObserver 클래스를 상속받는 클래스가 생성한 관찰자 객체를 Demultiplexer 객체에 연결하고 서버 측의 Observable 클래스를 상속받는 클래스가 생성한 피관찰자 객체를, Demultiplexer 객체와 연결된 Multiplexer 객체에 상호연결 하기만 하면 서버 측의 Observable 객체의 값이 변경될 때마다 클라이언트 측의 MobileObserver 객체가 변경된 값을 전달받는다. 따라서, 모바일 M/VC 응용 프레임워크는 무선 환경을 고려하지 않고 Multiplexer와 Demultiplexer 객체를 통한 조립식 연결만으로(구성적 인터페이스를 통해) 모바일 응용의 신속한 작성을 지원하여 응용 프로그램의 생산성을 향상시킨다.

2장에서는 모바일 관찰자/피관찰자 패턴에 대해 설명하고 3장에서는 모바일 M/VC 응용 프레임워크가 지원하는 구성적 인터페이스를 설명한다. 4장에서는 모바일 M/VC 응용 프로그램의 구현을 간략히 설명하고 모바일 댐 수위제어 응용에 적용한 예를 설명한다. 5장에서는 본 논문의 결론을 정리한다.

2. 모바일 관찰자/피관찰자 패턴

모바일 관찰자/피관찰자 패턴은 M/VC 기반의 모니터링 및 제어 응용에 자연스럽게 적용될 수 있다. 관찰자는 관찰하고자 하는 피관찰자에게 자신을 등록하고 피관찰자는 자신의 상태 변화를 등록된 각 관찰자에게 통보한다. 따라서, 모델에는 피관찰자의 역할을 배정하고 뷰·컨트롤러에는 관찰자의 역할을 배정한다. J2ME의 MIDP(Mobile Information Device Profile)[6, 8]는 자바의 Observer/Observable 클래스를 지원하지 않으므로 Observer 패턴을 무선 환경으로 확장한 MobileObserver 클래스를 정의하여 사용한다.

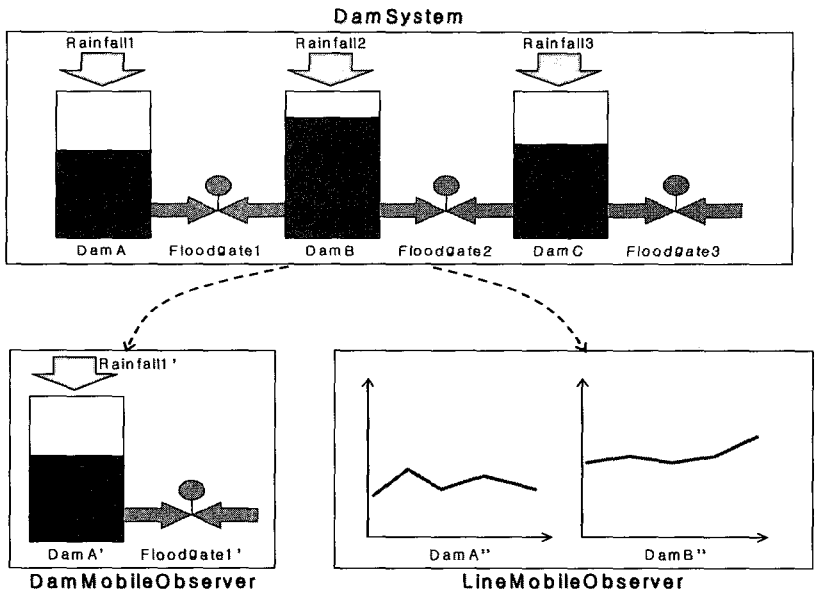


〈그림 2〉 모바일 관찰자/피관찰자

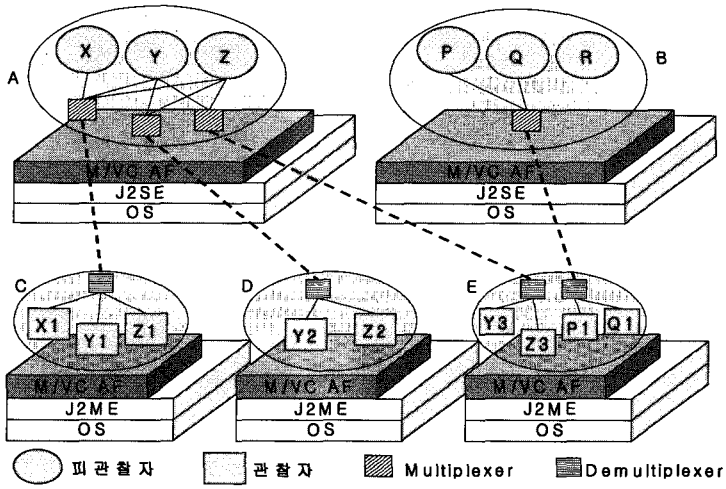
그림 2와 같이, 모바일 관찰자/피관찰자 간의 통신 및 상호작용은 논리적 레벨과 물리적 레벨로 나누어 볼 수 있다. 논리적 레벨에서, 모바일 관찰자/피관찰자는 자바의 관찰자/피관찰자와 같이 동작한다. 즉, 모바일 관찰자는 모바일 피관찰자들에게 자신을 등록하고 모바일 피관찰자는 자신의 상태가 변하면 이를 등록된 모바일 관찰자에게 통보한다. 물리적 레벨에서의 통신 및 상호작용은 서버 측의 Multiplexer와 클라이언트 측의 Demultiplexer에 의해 이루어진다. 서버 측의 Multiplexer는 관찰자로 구현되고 클라이언트

측의 Demultiplexer는 J2ME에 추가된 모바일 피관찰자 클래스로 구현된다. 따라서, 모바일 관찰자들은 피관찰자인 Demultiplexer에게 등록하고 Multiplexer는 모바일 피관찰자에게 등록하면 모바일 피관찰자들의 상태 변화는 등록된 Multiplexer 객체에게 통보되고 Multiplexer는 Demultiplexer에게 변화된 값을 전달한다. Demultiplexer는 전달받은 값을 등록된 모바일 관찰자들에게 통보한다.

그림 3의 모바일 댐 수위제어 응용은 모델의 역할을 담당한 DamSystem과 뷰·컨트롤러의 역할을 담당한 DamMobileObserver, LineMobileObserver로 구성된다. DamSystem은 DamA, DamB, DamC, Floodgate1, Floodgate2, Floodgate3, Rainfall1, Rainfall2, Rainfall3의 9개 피관찰자 객체들로 구성되고, DamMobileObserver는 DamSystem의 DamA, Floodgate1, Rainfall1 객체들을 관찰하는 관찰자 객체인 DamA', Floodgate1', Rainfall1'으로 구성되며, LineMobileObserver는 DamSystem의 DamA, DamB 객체를 꺾은선 그래프 형식으로 관찰하는 DamA'', DamB'' 관찰자 객체들로 구성된다. 클라이언트 측인 DamMobileObserver는 서버 측인 DamSystem의 DamA, Floodgate1, Rainfall1 객체를 모니터링 하기 위해 모바일 관찰자 객체인 DamA', Floodgate1', Rainfall1'을 Demultiplexer 객체에 연결한다. 같은 방식으로 LineMobileObserver의 모바일 관찰자 객체인 DamA''과 DamB''도 다른 Demultiplexer 객체에 연결한다. 서버 측인 DamSystem의 Multiplexer 객체들과 클라이언트 측의 Demultiplexer 객체들은 쌍으로 존재하며 일대일 관계로 연결된다. DamSystem의 피관찰자 객체들의 상태가 변하면 모든 피관찰자 객체들은 등록된 모든 Multiplexer들에게 값의 변경을 통지한다. Multiplexer 객체들은 새로운 상태를 전달받자마자 연결된 Demultiplexer 객체들에게 이를 전달한다. Demultiplexer 객체들은 새로운 상태를 전달받으면 변경된 값을 DamMobileObserver와 LineMobileObserver의 모바일 관찰자 객체인 DamA', Floodgate1', Rainfall1'과 DamA'', DamB''에게 통지한다.



〈그림 3〉 모바일 댐 수위제어 응용의 작성



〈그림 4〉 모바일 관찰자/피관찰자의 구조

이러한 모바일 관찰자/피관찰자 패턴을 무선 환경에 확장 적용하면 사용자 입출력과 인터페이스를 담당하는 모바일 관찰자(뷰·콘트롤러)는 클라이언트가 되고 데이터를 유지하는 피관찰자(모델)는 서버로 모델링할 수 있다. 클라이언트와 서버가 관찰자 피관찰자로서 동작하기 위해 필요한 무선 통신 및 상호작용은 모바일 M/VC 응용 프레임워크가 제공하는 Multiplexer와 Demultiplexer 클래스들에 의해 이루어진다. 서버 측의 피관찰자의 상태가 변하면 모든 클라이언트 측의 모바일 관찰자 객체들의 상태가 자동으로 변경되므로 정보의 변화를 장소의 제약 없이 신속하게 반영할 수 있다.

3. 모바일 M/VC 응용 프레임워크의 구성적[1] 인터페이스

본 절에서는 모바일 M/VC 응용 프레임워크를 이용한 응용 프로그램의 조립식 작성 과정을 설명한다. 모바일 M/VC 응용 프레임워크를 이용하여 모바일 응용을 작성하는 과정은 Observable과 MobileObserver 클래스들로부터 각각 필요한 모바일 관찰자/피관찰자 객체들을 생성한 후, 이들을 해당 Multiplexer 객체와 Demultiplexer 객체에 구성적으로 상호연결 하기만 하면 된다.

그림 4는 서버 측 A, B와 클라이언트 측 C, D, E가 분산되어 존재하는 M/VC 응용 프로그램을 모바일 M/VC 응용 프레임워크를 이용하여 작성하는 예이다. 클라이언트 C는 서버 A의 X, Y, Z 객체를 모니터링하기 위해 X1, Y1, Z1을 Demultiplexer 객체에 연결하고 서버 A의 피관찰자 객체 X, Y, Z는 Multiplexer 객체에 연결한다. Multiplexer 객체와 Demultiplexer 객체는 쌍으로 존재하며 일대일 관계로 연결된다. 다른 서버와 클라이언트 측의 관찰자/피관찰자 객체들도 이와 같은 plug-and-play 방식으로 상호연결 하여 응용 프로그램을 완성한다.

기존 객체지향 기술의 절차적 인터페이스에서는 개발자가 작은 단위의 프로시저들을 잘 이해하여 객체간의 필요한 상호작용을 프로그래밍해야 하는 반면, 모바일 M/VC 응용 프레임워크가 지원하는 구성적 인터페이스는 상호작용 패턴을 캡슐화 하여·조립식 연결만으로 모바일 응

1) '전체'를 창조해내기 위해 여러 요소들을 결합, 배치하여 하나의 예술작품을 성립시키는 방법. 모아서 조립한다는 의미.

용 프로그램을 개발할 수 있다. 즉, 무선 환경 하에서 plug-and-play식의 컴포넌트를 지원함으로써 컴포넌트 재사용성을 개선할 수 있고 소프트웨어의 생산성을 높일 수 있다.

객체 지향 시스템은 데이터와 행위를 캡슐화한 객체들로 구성되며 이 객체들의 상호작용으로 시스템이 운영된다. 그러나 구성 객체들을 상호작용 시키기 위해 프로그래머는 각 객체의 인터페이스(메소드들)를 이해하고 이를 프로그램에 적용해야 한다. 따라서, 구성 객체가 많을수록 이들을 제어하고 상호작용 시키기 위한 작업이 복잡해진다[9].

구성적 인터페이스는 객체간의 상호작용에 대한 세부사항은 고려할 필요 없이, 객체들을 상호 연결해 주기만 하면 동작하는 시스템을 지원한다. 구성적 인터페이스는 기존 객체지향 기술에서 제공하는 절차적 인터페이스보다 고급 인터페이스로서 그 개념을 그림 5의 연결되어 동작하는 팩스 기계의 하드웨어 예를 통해 설명한다.

〈그림 5〉 구성적 인터페이스

팩스 기계를 설치하기 위해 필요한 작업은 팩스 기계를 케이블(구성적 인터페이스)로 연결하기만 하면 된다. 이때, 케이블 내부의 각 선이 어떤 색이고 또 어떤 신호를 전달하는지에 대한 세부사항은 전혀 고려할 필요가 없다. 즉, 사용자는 연결되어 동작하는 두 대의 팩스 기계가 한 개의 논리적인 링크에 의해 연결된 것으로 간주할 수 있다. 상호작용 패턴을 캡슐화 하여 구성적 인터페이스를 지원하는 두 대의 팩스 기계는 케이블을 통한 연결만으로 적절히 동작한다.

구성적 인터페이스는 객체의 인터페이스를 하나의 단위로 간주하는 반면, 기존의 절차적 인터페이스는 객체의 인터페이스를 프로시저(메소드)들의 집합으로 간주한다. 따라서, 절차적 인터페이스를 이용하는 프로그래머는 프로그램에서 시스템을 구성하는 각 객체들 간의 상호작용을 메시지 전달을 통해 일일이 구현해야 한다. 반면, 구성적 인터페이스를 이용하는 프로그래머는 각 분산 객체들 간의 상호작용에 대한 세부사항(각 객체의 메소드들과 이들을 어떤 순서로 작동시킬 것인지에 대한 사항)을 고려하지 않고 분산 객체들을 구성적으로 연결하기만 하면 된다. 즉, 구성적 인터페이스를 지원하는 모바일 객체간의 조립식 연결만으로 모바일 응용을 쉽고 신속하게 개발할 수 있다.

표 1은 관찰자/피관찰자 패턴을 이용하고 Multiplexer와 Demultiplexer 클래스를 지원하여 객체지향 기술에 조립성을 추가한 모바일 M/VC 응용 프레임워크와 CherryPy 2.1[10], helma 1.4.3[11], Echo 1.0.5[12] 등의 기존 객체지향 기술을 기반으로 하는 응용 프레임워크의 기능을 비교한 것이다.

CherryPy는 Python 언어를 지원하는 웹 응용 프레임워크로서 객체지향 기술을 기반으로 하여 간결한 웹 및 모바일 응용 프로그램을 신속하게 생성할 수 있다. helma는 웹 사이트와 인터넷 응용을 신속하고 효율적으로 생성할 수 있는 공개된 웹 응용 프레임워크이다. Echo는 객체지향이면서 사건중심인(event-driven) 웹 및 모바일 Java 응용의 신속한 작성을 지원하는 프레임워크다. 그러나 이러한 세 프레임워크는 절차적 인터페이스를 기반으로 하는 기존의 객체지향 기술을 바탕으로 하기 때문에 프로그래머는 서비스 컴포넌트 객체들의 메소드를 이해하고 정해진 절차를 적용해야 한다. 반면, 모바일 M/VC 응용 프레임워크가 제공하는 구성적 인터페이스는 모바일 객체 간의 상호작용 패턴을 클래스 내로 캡슐화하여 구성적 연결만으로 모바일 응용 프로그램을 쉽게 개발할 수 있게 한다. 즉, 모바일 환경 하에서 plug-and-play 방식의 컴포넌트를 지원함으로써 컴포넌트의 재사용성을 개

〈표 1〉 객체지향 기술 기반의 응용 프레임워크와 모바일 M/VC 응용 프레임워크

	객체지향 기술 기반 응용 프레임워크(CherryPy 2.1, helma 1.4.3, Echo 1.0.5)	모바일 M/VC 응용 프레임워크
응용 분야	일반	실시간 모니터링 및 제어
인터페이스	절차적	구성적
조립성	+	+++
프로그램 개발 속도	+	+++
재사용성	+	++
확장성	+	++
코드 크기	++	+

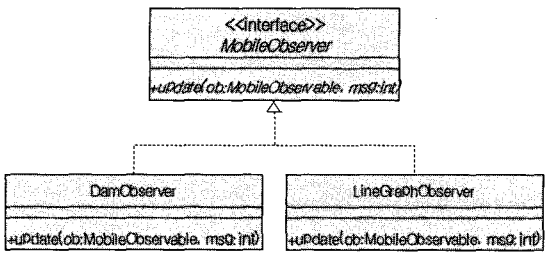
선할 수 있고 소프트웨어의 생산성 및 확장성을 향상시킬 수 있다.

4. 모바일 M/VC 응용 프레임워크의 구현 및 적용

본 절에서는 모바일 M/VC 응용 프레임워크의 구성요소를 간략히 설명하고 모바일 M/VC 응용 프레임워크로 작성된 모바일 댐 수위제어 응용의 동작원리를 설명한다. 서버는 J2SE 플랫폼을 사용하고 클라이언트는 무선기기 이용을 위해 J2ME 플랫폼을 사용한다.

4.1 모바일 M/VC 응용 프레임워크의 구성요소

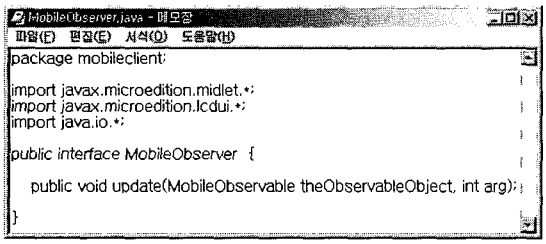
모바일 M/VC 응용 프레임워크는 관찰자/피관찰자 패턴을 이용하여 구현된다. J2SE 플랫폼은 관찰자/피관찰자 패턴을 이용하기 위한 Observer /Observeble 클래스를 지원하나 J2ME는 지원하지 않으므로 클래스를 정의하여 모바일 M/VC 응용 프레임워크에 추가한다. 모바일 M/VC 응용 프레임워크는 Multiplexer, Demultiplexer, MobileObserver, MobileObservable 클래스들을 포함한다. 서버 측은 J2SE 플랫폼 상에서 J2SE의 Observable 클래스와 모바일 M/VC 응용 프레임워크의 Multiplexer 클래스를 상속받아 활용한다. 모바일 클라이언트 측은 M/VC 응용 프레임워크의 MobileObserver, MobileObservable, Demultiplexer 클래스들을 사용한다. 즉, MobileObserver 클래스를 상속받는 자식 클래스의 객체를 생성하여 이를 Demultiplexer 객체와 plug-and-play식의 조립식 연결을 통해 클라이언트 측 응용이 작성된다.



〈그림 6〉 MobileObserver 클래스의 계층구조

그림 6은 MobileObserver 클래스를 활용한 계층구조이다. 피관찰자는 하나이지만 관찰자는 다수일 수 있으며 GUI적인 뷰의 형태도 다양하다. 뷰의 모습은 다르지만 관찰되는 값은 같으므로 MobileObserver 인터페이스를 정의하여 피관찰자들이 인터페이스를 클래스로 구현하도록 한다. 그림 7의 MobileObserver 인터페이스는 자신을 구현하는 클래스들에게 update (Mobile-

Observable ob, int msg)라는 메소드를 구현하도록 요구하게 된다. 첫 번째 인자는 피관찰자 객체이고 두 번째 인자는 부가 정보이다. 모델의 값이 변경되면 소켓을 통하여 모든 관찰자들에게 통지되고 관찰자들의 update 메소드가 호출된다.



〈그림 7〉 MobileObserver 인터페이스

그림 8은 MobileObservable 클래스를 정의한 것이다. MobileObservable 클래스는 J2SE의 Observable 클래스와 마찬가지로 아홉 개의 메소드를 갖는다. Demultiplexer 객체의 run 메소드에서 MobileObservable 클래스로부터 상속된 setChanged 메소드가 호출되어 현재 관찰되고 있는 관찰자 객체의 값이 변경된다. 그리고 MobileObservable 클래스로부터 상속된 notifyMobileObservers 메소드가 호출되어 등록된 모든 피관찰자 객체가 update 메소드를 호출하도록 하여 출력 결과를 생성한다.

Demultiplexer 객체는 모바일 클라이언트 측에서 사용되는 Observable 클래스이다. 서버 측에서 접속을 요청하면, Demultiplexer 객체를 위해 생성된 Multiplexer 객체가 연결된다. Demultiplexer 객체는 서버 측에 존재하는 분산된 피관찰자들의 값을 동시에 받는 Multiplexer 객체와 통신한다. Demultiplexer 객체는 분산된 피관찰자의 변경된 값을 받았을 때, 자신에 연결된 관찰자들에게 그 값을 전달한다.

Multiplexer 객체는 서버 측에서 사용되는 관찰자 클래스로서 자신에 연결된 피관찰자들 각각에 자신을 자바의 관찰자로 등록함으로써 피관찰자들을 모니터링 한다. 연결된 피관찰자의 상태가 변경되면 그 변경된 값을 받게 되고 자신과 일대일로 연결된 Demultiplexer 객체에게 이를 전달한다.

〈그림 8〉 MobileObservable 클래스

모바일 관찰자/피관찰자 응용은 관찰자가 피관찰자를 관찰하기 위해 자신을 등록하는 경우와 피관찰자의 상태가 변경될 때 이를 관찰자에게 통지하는 경우의 두 가지 경로가 존재한다.

클라이언트 측의 모바일 관찰자는 서버 측의 피관찰자를 관찰하기에 앞서 다음과 같은 순서로 등록한다.

1. 모바일 관찰자는 Demultiplexer 객체의 addObserver(Observer observe, int objIDX) 메소드를 호출하여 자신을 Demultiplexer 객체를 관찰하는 관찰자로 등록한다. objIDX는 무선 피관찰자를 지정하기 위한 객체 색인으로 클라이언트 측에서 사용된다.
2. Demultiplexer 객체는 관찰자를 관찰자 목록

에 더하고 서버 측의 Mutliplexer 객체에게 ObjIDX를 포함하는 REGISTER 메세지를 보낸다.

3. Multiplexer 객체가 objIDX가 포함된 REGISTER 메세지를 받으면 objIDX로 지정된 피관찰자의 객체 참조를 찾고 분산된 피관찰자의 addObserver(Observer mpx) 메소드를 호출한다.

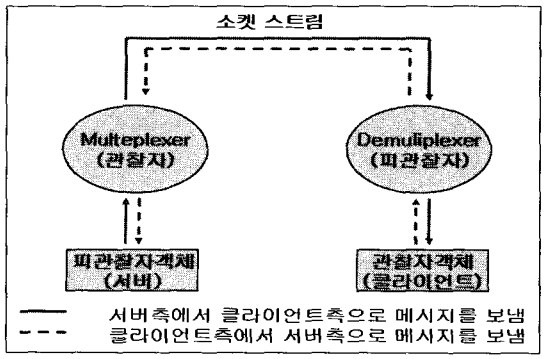
서버 측의 피관찰자는 상태가 변할 때 클라이언트 측의 모바일 관찰자에게 다음과 같은 순서로 통지한다.

1. 서버 측의 피관찰자는 notifyObservers(Object newState) 메소드를 호출함으로써 등록된 모든 Multiplexer 객체들에게 값의 변경을 통지한다. 매개변수 newState는 객체 색인 objIDX와 피관찰자의 새로운 값 newVal로 조합된 문자열이다.
2. Multiplexer 객체는 newState를 전달받자마자 연결된 Demultiplexer 객체에게 이를 전달한다.
3. Demultiplexer 객체는 newState를 전달받아 objIDX와 newValue를 추출하고 notifyObservers(Object newVal) 메소드를 호출하여 클라이언트 측의 모든 모바일 관찰자들에게 통지한다.

4.2 모바일 M/VC 응용 프레임워크를 이용한 댐 수문제어 응용

본 절에서는 모바일 M/VC 응용 프레임워크를 이용하여 구현한 응용의 동작원리를 소개하고 댐 수문제어 응용에 적용한 결과를 보인다.

그림 9는 모바일 M/VC 응용 프레임워크로 작성한 모바일 M/VC 응용의 객체들 간의 메시지 흐름도이다. 서버 측에는 Multiplexer 객체를 부착하고 클라이언트 측에는 Demultiplexer 객체를 부착하여 일대일로 물리적인 연결을 구성



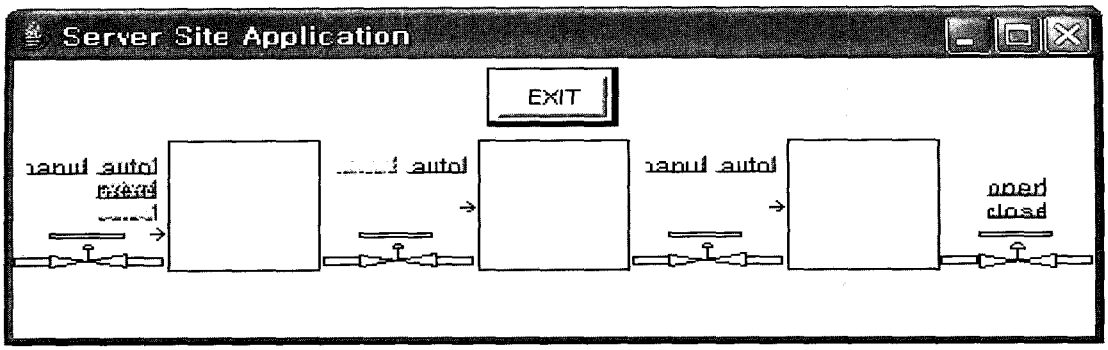
〈그림 9〉 모바일 M/VC 응용의 객체들 간의 메시지 흐름도

하여 논리적인 연결을 지원하며 연결은 소켓을 통해서 이루어진다. 서버 측의 피관찰자 객체들은 Multiplexer 객체와 연결되고 클라이언트 측의 관찰자 객체들은 Demultiplexer 객체와 연결된다.

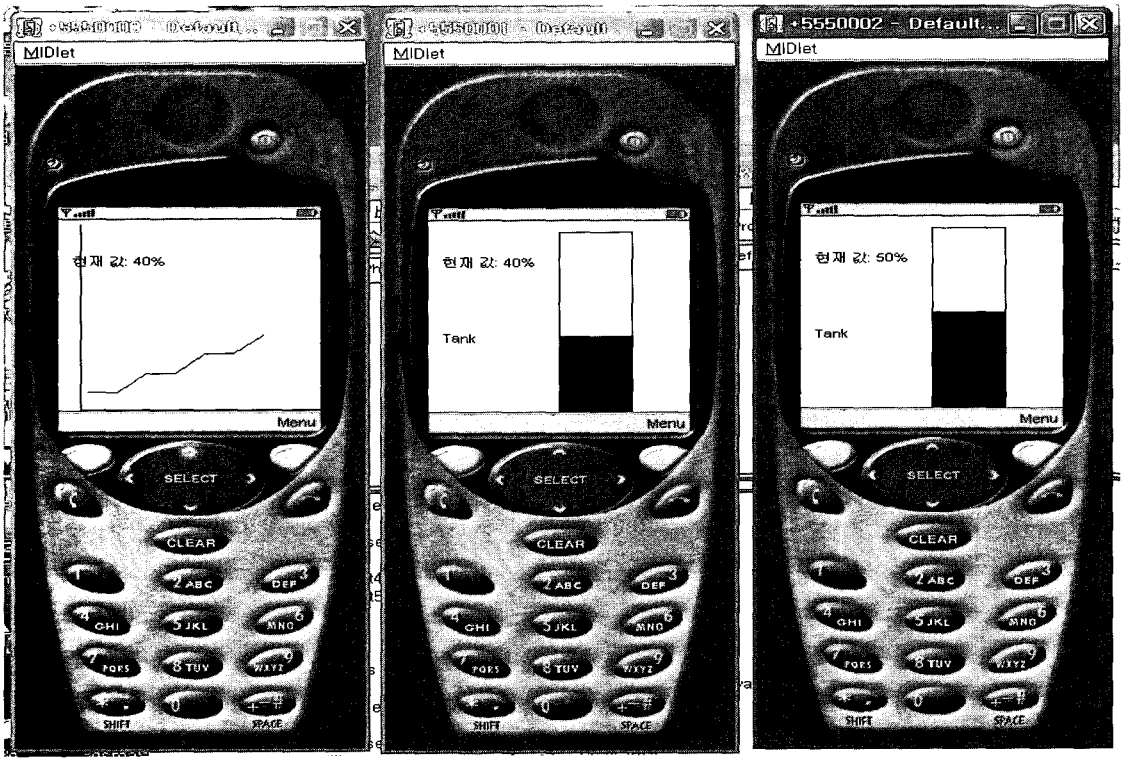
Multiplexer 객체는 서버 측에 존재하는 관찰자이며 피관찰자를 모니터링하고 피관찰자에게 스스로를 관찰자로서 등록한다. Multiplexer 객체가 피관찰자로부터 새로운 값을 받으면 이를 클라이언트와 연결된 Demultiplexer 객체에게 전송한다. Demultiplexer 객체는 클라이언트 측에 존재하는 피관찰자이며 처음 생성될 때 서버 측에 소켓 연결을 요청하여 Multiplexer 객체와 통신한다. Demultiplexer 객체가 변경된 값을 전달받으면 클라이언트 측의 관찰자들에게 변경된 값을 통지한다.

댐 수문제어 응용은 간단한 모니터링 및 제어 응용으로서 서버 측 시스템에는 그림 10과 같이 3개의 댐과 댐 사이의 흐름을 제어하는 4개의 밸브를 가진다. 댐 수문제어 응용의 피관찰자 객체는 댐과 4개의 밸브이다. 그림 11은 3개의 댐 수위의 변화를 실시간으로 디스플레이 하는 3개의 서로 다른 관찰자들의 캡쳐 화면이다.

DamSystem을 포함하는 서버 측 응용은 ServerSiteApp 클래스의 객체이며 ServerSiteApp 클래스는 자바 AWT 패키지의 Frame



〈그림 10〉 댐 수문 제어 응용의 서버 측 시스템



〈그림 11〉 모바일 관찰자 I, II, III

클래스의 서브클래스이고, TableOfObservables, MasterServerThread, TankSystem 컴포넌트 등을 포함한다.

5. 결 론

현재, 무선기기 사용자가 급증하고 있다. 이러한 추세에 맞추어 지리정보, 교통안내 시스템과 같은 모니터링 및 제어 응용이 무선기기에 신속하게 지원되어야 한다. 본 논문에서는 모바일 응용 중, 모니터링 및 제어 응용인 모바일 M/VC 응용 프로그램의 신속한 조립식 작성을 지원하는 모바일 M/VC 응용 프레임워크를 소개하였다.

모바일 M/VC 응용 프레임워크를 이용한 개

발자는 무선 환경을 고려하지 않고 필요한 모바일 관찰자 객체와 피관찰자 객체들을 생성하여 모바일 피관찰자는 서버 측의 Multiplexer 객체에, 모바일 관찰자는 클라이언트 측의 Demultiplexer 객체에 plug-and-play 방식으로 조립하기만 하면 응용 프로그램이 생성된다. 따라서, 모바일 M/VC 응용 프레임워크는 무선 환경에서 컴포넌트 재사용성을 개선하고 신속한 모바일 M/VC 응용의 개발을 지원하여 소프트웨어의 생산성을 향상시킨다. 표 1에는 기존 객체지향 기술 기반의 응용 프레임워크와 모바일 M/VC 응용 프레임워크의 기능을 비교하였다.

참 고 문 헌

[1] 유무선 통신서비스 가입자 현황(2004년 10월말 통계), http://mic.go.kr/.

[2] Michael R Blaha and James R Rumbaugh, Object-Oriented Modeling and Design with UML 2nd Ed., Prentice Hall, November 2004.

[3] Krzysztof Cwalina and Brad Abrams, Framework Design Guidelines, Addison-Wesley, September 2005.

[4] Partha Kuchana, Software Architecture Design Patterns in Java, Auerbach Publications, April 2004.

[5] Martyn Mallick, Mobile and Wireless Design Essentials, WILEY, 2003.

[6] de Jode, M, Programming the Java 2 micro edition for Symbian OS: a developer's guide to MIDP 2.0, John Wiley & Sons, 2004.

[7] Daryl Wilding-McBride, Java Development on PDAs: Building Applications for Pocket PC and Palm Devices, Addison- Wesley, June 2003.

[8] Roger Riggs, Antero Taivalsaari, Jim Van Peursem, Jyri Huopaniemi, Mark Patel and Aleksi Uotila, Programming Wireless Devices with the Java™2 Platform, Micro Edition, Addison-Wesley, 2003.

[9] P. Clements and Linda Northrop, Software Product Lines-Practice and Patterns, Addison-Wesley, 2002.

[10] CherryPy 2.1, http://www.cherrypy.org/.

[11] helma 1.4.3, http://helma.org/.

[12] Echo 1.0.5, http://freshmeat.net/projects/echo/.

[13] D. Batory, Feature-Oriented Programming and the AHEAD Tool Suite, Proceedings of the 26th International Conference on Software Engineering(ICSE), Edinburgh Scotland, May 2004.

[14] Chi-Wei Lan, Chun-Chou Chien, Meng-Yen Hsieh, and Irene Chen, A Mobile e-Commerce Solution, Proceedings of the International Symposium on Multimedia Software Engineering, pp. 215-222, 2000.

[15] Doohun Eum and Toshimi Minoura, WebSiteGen: Web-Based Database Application Generator, IEICE Trans. on Information & Systems, Vol. E86-D, No. 6, pp. 1001-1010, June 2003.

[16] Miguel P. Monterio and Joao M. Fernandes, Towards a catalog of aspect-oriented refactoring, Proceedings of the 4th International Conference on Aspect-oriented Software Development, Chicago Illinois, pp. 111-122, 2005.

◐ 저 자 소 개 ◑

음 두 헌 (Doohun Eum)
1984년 서강대학교 전자공학과 졸업(학사)
1987년 오레곤주립대학교 대학원 컴퓨터공학과 졸업(석사)
1990년 오레곤주립대학교 대학원 컴퓨터공학과 졸업(박사)
1991～1992 전자통신연구원 인공지는연구실 선임연구원
1992～현재 덕성여자대학교 컴퓨터공학부 교수
관심분야 : 객체지향 시스템, 분산 및 모바일 응용, etc.
E-mail : dheum@duksung.ac.kr