모바일 M/VC 응용 프레임워크*

강이지°, 박은희, 음두헌
덕성여자대학교 전산학과
izzy@duksung.ac.kr°, skylove102102@hotmail.com, dheum@duksung.ac.kr

Mobile M/VC Application Framework

Izzy Kang°, Eunhee Park, Doohun Eum
Dept. of Computer Science, Duksung Women's Univ.

요약

최근 무선 기기 사용자가 급증하고 있다. 이에 따라 교통제어 시스템과 같은 모니터링 및 제어 응용이 무선 기기에서 활발히 사용될 것으로 예상된다. 본 논문에서는 모바일 응용 중 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 Multiplexer, Demultiplexer 클래스들을 지원한다. 개발자는 이 프레임워크를 이용하여 Observable과 MobileObserver 클래스들로부터 필요한 객체를 생성한 후, 이들을 Multiplexer와 Demultiplexer 클래스가 생성한 객체에 구성적으로 상호 연결하여 응용을 생성한다. 즉, 개발자는 무선 환경을 고려하지 않고 모바일 M/VC 응용 프레임워크가 제공하는 Multiplexer나 Demultiplexer 클래스의 객체에 무선 관찰자/피관찰자 객체들을 조립식으로 연결함으로써 피관찰자의 상태 변화가 관찰자에게 전달되고, 관찰자를 통한 사용자의 입력이 피관찰자에게 전달되어 반영되는 모바일 응용을 신속하게 생성할 수 있다. 따라서, 모바일 M/VC 응용 프레임워크는 무선 통신 환경하의 컴포넌트 재사용성을 개선하고, 모바일 M/VC 응용의 생산성을 향상시킨다.

1. 서론

무선 기기 사용자가 폭발적으로 증가함에 따라 생산 공정 시스템, 교통 제어 시스템 등과 같은 모니터링 및 제어 응용이 무선 기기에서 활발히 사용될 것으로 기대된다. 소비자가 직접 구매하는 장소인 도매점을 여러 개 관리하는 회사와 물품들의 재고를 비축해 놓고 판매하는 도매점들로 구성된 생산 공정 시스템의 경우, 회사는 도매점들의 재고량 수준을 모니터링하고, 공급자에 대해서는 가격을 기준으로 적절한 공급자를 결정한 후, 물품 주문을 제어해야 한다[1]. 택시 회사와 같은 곳에서도 모니터링 및 제어 응용을 도입한 교통 제어 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 모니터링하고, 수문을 열고 닫음을 제어할 수 있다. 이러한 모니터링 및 제어 응용에 무선 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 있다. GUI(Graphical User Interface) 시스템이나 클라이언트/서버 시스템처럼 복잡한 기능을 갖는 객체들을 분류하는 방법론 중 하나인 MVC(Model, View, Controller) 패턴을 기반으로 모바일 응용의 생산성을 향상시키고자 하는 것이 모바일 M/VC 응용 프레임워크의 목표이다. 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 상호 작용함으로써 수행된다. 여기서, 뷰와 컨트롤러는 일반적으로 GUI에 해당되는 요소들로서 결합되어 제공되므로, MVC 모델을 M/VC 모델로 표기할 수 있다. 또한, 객체들 간의 상호작용을 자동 처리하기 위해, Java의 관찰자/피관찰자(Observer/Observable)[3]를 무선 관찰자/피관찰자 패턴으로 확장한다. 모바일 M/VC 응용의 신속한 작성을 위해 모바일 M/VC 응용 프로그램의 개발에 공통적으로 사용될 수 있는 구성 요소인 Multiplexer와 Demultiplexer, MobileObserver/Observable과 이들의 구조를 일반화하여 프레임워크를 구성한 것이 모바일 M/VC 응용 프레임워크(Mobile M/VC Application Framework)이다.

그림 1은 모바일 M/VC 응용 프레임워크를 이용하여 생성한 모바일 응용의 구조이다.

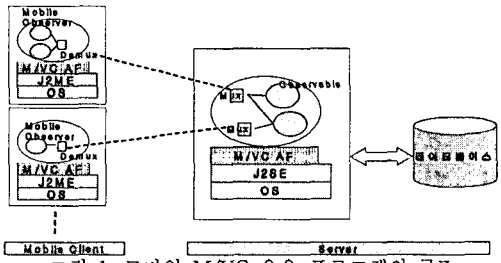

그림 1. 모바일 M/VC 응용 프로그램의 구조

모바일 클라이언트 측은 무선 기기의 운영체제 위에 J2ME 플랫폼이 위치한다. J2ME 플랫폼은 사용되는 무선 기기별로 Java 플랫폼을 정의하는 프로파일을 제공하여 개발과 실행을 가능하게 한다. J2ME 플랫폼 위에 모바일 M/VC 응용 프레임워크가 위치한다. M/VC 응용 프레임워크는 Multiplexer와 Demultiplexer, MobileObserver 클래스 등을 포함한다. 모바일 클라이언트 측에서는 MobileObserver와 Demultiplexer 클래스를 상속받는 클래스를 생성하여 모바일 응용을 작성한다. 서버 측에는 운영체제 위에 J2SE 플랫폼이 위치하고, 그 위에 M/VC 응용 프레임워크가 위치한다. M/VC 응용 프레임워크의 Demultiplexer 클래스를 상속받는 클래스와 J2SE가 제공하는 Observer/Observable 클래스의 객체들을 생성하여 서버 응용이 작성된다.

서버 측의 Observable 클래스가 생성한 피관찰자 객체를 모니터링하기 위해 모바일 클라이언트 측의 MobileObserver 클래스를 상속받는 클래스가 생성한 객체를 Demultiplexer 객체에 연결하고, 서버 측의 Observable클래스가 생성한 피관찰자 객체를 Multiplexer 객체에 상호연결하기만 하면, 서버 측의 Observable 객체의 값이 변경될 때마다 클라이언트 측의 MobileObserver 객체가 변경된 값을 전달받는다. 따라서, 모바일 M/VC 응용 프레임워크는 무선 환경을 고려하지 않은

* 본 연구는 학국과학재단 목적기초연구 (R06-2002-003-01004-0(2004)) 지원으로 수행되었음.

조립식 연결만으로 모바일 응용의 신속한 작성을 지원하여 응용 프로그램의 생산성을 향상시킬 뿐만 아니라 기존 프로그램들이 일정한 장소에서만 사용 가능하여 이동성에 제약을 받았던 문제들을 해결할 수 있다.

모바일 응용의 구조에는 무선 기기에 마이크로 브라우저만 요구되는 얇은 클라이언트(thin client) 구조와 J2ME와 같은 프로그램 실행 환경이 요구되는 스마트 클라이언트(smart client) 구조의 2가지가 있다[4]. 얇은 클라이언트 구조는 서버 측과의 무선 연결이 항상 유지되어야하며 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.

2절에서는 무선 관찰자/피관찰자 패턴에 대해 설명하고, 3절에서는 모바일 M/VC 응용 프로그램의 조립식 작성을 설명한다. 4절에서는 모바일 M/VC 응용 프로그램의 동작원리를 모바일 댐 수위 제어 응용이라는 예제를 중심으로 설명한다. 5절에서는 본 논문의 결론을 정리한다.

2. 무선 관찰자/피관찰자 패턴

무선 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 모델에 피관찰자를, 뷰·컨트롤러에 관찰자를 적용한다. MIDP(Mobile Information Device Profile)에서는 자바의 Observer/Observable 객체를 사용할 수 없으므로 Observer 패턴을 확장한 MobileObserver 클래스를 정의하여 사용한다.

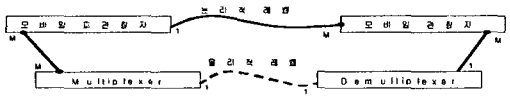

그림 2. 무선 관찰자/피관찰자

그림 2와 같이, 무선 관찰자/피관찰자 간의 통신 및 상호 작용은 논리적 레벨과 물리적 레벨로 나누어 볼 수 있다. 논리적 레벨에서, 무선 관찰자/피관찰자는 자바의 관찰자/피관찰자와 같이 동작한다. 즉, 무선 관찰자는 무선 피관찰자에게 자신을 등록하고 무선 피관찰자는 자신의 상태가 변하면 이를 등록된 무선 관찰자에게 알려 준다. 물리적 레벨에서의 통신 및 상호 작용은 서버 측의 Multiplexer와 클라이언트 측의 Demultiplexer에 의해 이루어진다. 서버 측의 Multiplexer는 관찰자로 구현되고, 클라이언트 측의 Demultiplexer는 J2ME에 무선 피관찰자 클래스를 추가하여 이를 이용한다. 따라서 무선 관찰자들은 피관찰자인 Demultiplexer에게 등록하고, Multiplexer는 무선 피관찰자에게 등록한다. 무선 피관찰자들은 상태가 변하면 등록된 Demultiplexer에게 이를 알려 주고, Multiplexer는 Demultiplexer에게 변화된 값을 전달한다. Demultiplexer는 전달받은 값을 등록된 무선 관찰자에게 알려 준다.

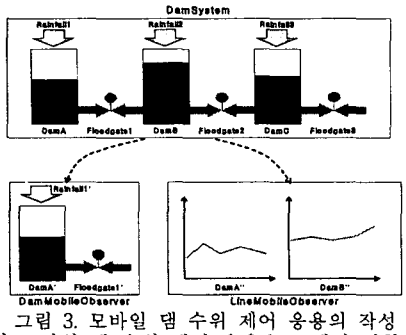

그림 3. 모바일 댐 수위 제어 응용의 작성

그림 3의 모바일 댐 수위 제어 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 모니터링하기 위해 무선 관찰자 객체인 DamA', Floodgate1', Rainfall1'을 Demultiplexer 객체에 연결한다. 같은 방식으로 LineMobileObserver의 무선 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 전달한다. Demultiplexer들은 새로운 상태를 전달받으면 변경된 값을 DamMobileObserver와 LineMobileObserver의 무선 관찰자 객체인 (DamA', Floodgate1', Rainfall1')과 (DamA", DamB")에게 통지한다.

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

3. 모바일 M/VC 응용 프로그램의 조립식 작성

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 필요한 무선 관찰자/피관찰자 객체들을 생성한 후, 이들을 해당 Multiplexer 객체와 Demultiplexer 객체에 구성적으로 상호연결하기만 하면 된다. 그림 3의 모바일 댐 수위 제어 응용에서는 서버 측의 3개의 Dam 객체와 3개의 Floodgate 객체, 3개의 Rainfall 객체를 생성하여 Demultiplexer 객체에 연결한다. 클라이언트 측에서는 서버 측 객체들 중 모니터링 하고자 하는 객체들을 생성하여 Multiplexer 객체에 연결한다. 서버 측 Dam의 수위가 변경되면 클라이언트 측 Dam의 수위도 변경되고, 클라이언트 측에서 사용자가 값을 변경하면 서버 측 Dam의 값이 변경된다.

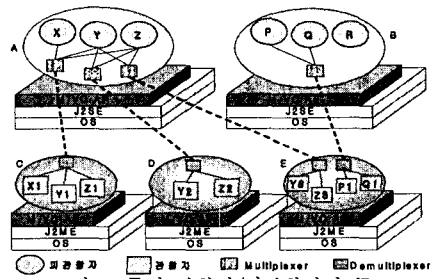

그림 4. 무선 관찰자/피관찰자의 구조

그림 4는 서버 측 (A, B)와 클라이언트 측 (C, D, E)이 분산되어 존재하는 보다 복잡한 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 Deultiplexer 객체에 연결하고, 서버 A의 피관찰자 객체 X, Y, Z는 Multiplexer 객체에 연결한다. Multiplexer 객체와 Demultiplexer 객체는 쌍으로 존재하며, 일대일 관계로 연결된다.

기존 객체지향 기술의 절차적 인터페이스(Procedural Interface)에서는 개발자가 작은 단위의 프로시저들을 잘 이해하여 객체간의 필요한 상호작용을 프로그래밍해야 하는 반면, 모바일 M/VC 응용 프레임워크가 지원하는 구성적 인터페이스(Structural Interface)는 상호작용 패턴을 캡슐화하여 조립식 연결만으로 모바일 응용 프로그램을 개발할 수 있다. 즉, 무선 환경 하에서 plug-and-play식의 컴포넌트를 지원함으로써 컴포넌트 재사용성을 개선할 수 있고 소프트웨어 생산성을 높일 수 있다.

4. 모바일 M/VC 응용 프로그램의 동작원리

본 절에서는 모바일 M/VC 응용 프레임워크로 작성된 모바일 댐 수위 제어 응용의 동작원리를 설명한다. 서버는 업무용 서버·시스템에 적합한 J2SE 플랫폼을 사용하고 클라이언트는 무선 기기 이용을 위해 J2ME 플랫폼을 사용한다.

M/VC 모델은 관찰자/피관찰자 패턴을 이용하면 쉽게 구현될 수 있다. J2SE 플랫폼에서는 관찰자/피관찰자 패턴을 이용하기 위한 Observer/Observeble 클래스를 지원하나 J2ME에서는 지원되지 않으므로 직접 클래스를 정의하여 모바일 M/VC 응용 프레임워크에 추가한다. 모바일 M/VC 응용 프레임워크는 MobileObserver, Multiplexer, Demultiplexer 클래스들로 구성된다. 서버 측은 J2SE 플랫폼 상에서 J2SE의 Observable 클래스와 모바일 M/VC 응용 프레임워크의 Multiplexer를 상속받고, 클라이언트 측의 J2ME 플랫폼에서는 모바일 M/VC 응용 프레임워크의 MobileObserver와 Demultiplexer 클래스를 상속받아 클래스들을 정의한다. 이러한 클래스들의 객체들로 모바일 M/VC 응용을 구성하게 된다.

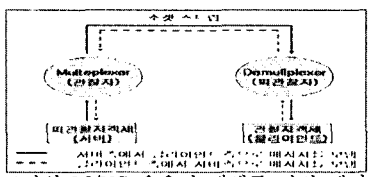

그림 5. 모바일 M/VC 응용의 객체들 간의 메시지 흐름도

그림 5는 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 구성하여 논리적인 연결을 지원하며 연결은 소켓을 통해서 이루어진다. 서버 측의 피관찰자 객체는 Multiplexer 객체와 연결되고 클라이언트 측의 관찰자 객체는 Demultiplexer 객체와 연결된다.

Multiplexer 객체는 서버 측에 존재하는 관찰자이며 피관찰자를 모니터링하고 피관찰자에게 스스로를 관찰자로서 등록한다. Multiplexer 객체가 피관찰자로부터 새로운 값을 받으면 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 통지한다.

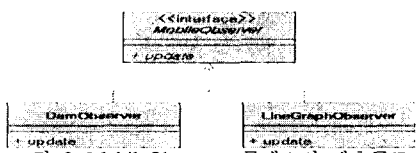

그림 6. MobileObserver 클래스의 계층구조

그림 6는 MobileObserver 클래스의 계층구조이다. 피관찰자는 하나이지만 관찰자는 다수일 수도 있으며 GUI적인 뷰의 형태도 다양하다. 뷰의 모습은 다르지만 관찰되는 값은 같으므로 MobileObserver 인터페이스를 정의하여 피관찰자들이 인터페이스를 상속받아 클래스를 구현하도록 한다. 이 인터페이스는 자신을 구현하는 클래스들에게 다음과 같은 update(Observable o, object arg)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.

```
/* 관찰자를 표현하는 인터페이스*/
interface Observer {
  public abstract void update(Observable o, object arg);
}
```

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 무선 관찰자는 서버 측의 피관찰자를 관찰하기에 앞서 다음과 같은 순서로 등록한다.

1. 무선 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 목록에 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 무선 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 전달한다.
3. Demultiplexer 객체는 newState를 전달받아 objIDX와 newValue를 추출하고 notifyObservers(Object newVal)메소드를 호출하여 클라이언트 측의 모든 무선 관찰자들에게 통지한다.

5. 결론

현재 무선 기기 사용자가 급증하고 있다. 이러한 추세에 맞추어 모니터링 및 제어 응용이 무선 기기에 신속하게 지원될 것으로 예상된다. 본 논문에서는, 모바일 응용 중, 모니터링 및 제어 응용인 모바일 M/VC 응용 프로그램의 조립식 작성을 지원하는 모바일 M/VC 응용 프레임워크를 소개하였다.

모바일 M/VC 응용 프레임워크를 이용한 개발자는 무선 환경을 고려하지 않고 필요한 무선 관찰자 객체와 피관찰자 객체들을 생성하여 무선 피관찰자는 서버 측의 Multiplexer 객체에 무선 관찰자는 클라이언트 측의 Demultiplexer 객체에 plug-and-play 방식으로 조립하기만 하면 된다. 따라서 모바일 M/VC 응용 프레임워크는 무선 환경에서 컴포넌트 재사용성을 개선하고, 신속한 모바일 M/VC 응용의 개발을 지원하여 소프트웨어의 생산성을 향상시킨다.

참고문헌

[1] 음두헌, 분산 객체 조립기를 이용한 MVC 응용의 구성적 작성 : 정보과학회 논문지(B) 제 26권 11호, 1999.11. pp. 1298-1305
[2] Stephen A. Stelting, Olav Maassen, Applied Java Patterns, Sun, 2002
[3] Budd, T., Understanding object-oriented programming with JAVA, Addison-wesley, 1998
[4] Martyn Mallick, Mobile and Wireless Design Essentials, WILEY, 2003