DOI: 10.3745/KIPSTD.2008.15-D.3.327

J2ME MIDlet 사용자 인터페이스 자동생성을 위한 XML언어

박 기 창[†] · 서 성 채[††] · 김 병 기[†††]

요 약

XML을 이용한 사용자 인터페이스(UI : User Interface)명세에 관한 연구로 다양한 UI 명세언어(UIDL : User Interface Description Language)들이 등장하였다. 기존의 UIDL들은 웹과 데스크톱 어플리케이션의 UI 명세를 지원하지만, 모바일 어플리케이션을 위한 명세는 지원하지 않는다. 본 논문에서는 자바의 모바일 플랫폼인 J2ME(Java Platform, Micro Edition)의 응용 모델인 MIDlet UI 명세를 위한 MIML(Midlet Interface Markup Language)을 제안한다. 제안한 MIML로부터 MIDlet의 UI와 관련된 자바 코드를 자동으로 생성하기 위한 규칙을 제안하고, 이를 이용하여 MIML을 자동으로 자바코드로 만들어주는 J2MERenderer를 구현하였다. 제안한 MIML은 모바일 UI 명세의 효율성을 제공하고, J2MERenderer는 모바일 어플리케이션의 UI 개발에 생산성과 일관성을 유지하도록 도와준다.

키워드 : 사용자 인터페이스 명세 언어, 코드자동생성, J2ME

XML Language for Generating J2ME MIDlet User Interfaces

Park Ki Chang[†] · Seong Chae Seo[††] · Byung Ki Kim[†††]

ABSTRACT

Many XML-compliant UIDLs(User Interface Description Languages) have been developed to specify user interfaces. Although previous UIDLs are helpful to describe user interfaces of web and desktop applications, they are not available of developing UI of mobile applications. In this paper, in order to effectively specify user interfaces of MIDlet which is application model on J2ME(Java Platform, Micro Edition), we propose MIML(Midlet Interface Markup Language) and present transformation rules to generate source codes from MIML. Further, we enhance the usability of MIML through J2MERenderer which is a tool using presented rules. The proposed method can specify user interfaces of mobile applications and allow developers to keep productivity and consistency in development phase.

Key Words : User Interface Description Language, Code Generation, J2ME

1. 서 론

클라이언트 환경이 다양화됨에 따라 모바일 프로그램에 대한 수요가 증가하고 있다. 모바일 프로그램이 탑재되는 장치로는 RIM(Research In Motion), Cellular Telephone, PDA, Tablet PC, Laptop PC등이 있으며, 이러한 장치에서 동작하는 모바일 플랫폼으로는 대표적으로 J2ME(Java2 Micro Edition)가 있다. J2ME는 한정된 메모리, 디스플레이 장치, 전원 장치 등의 특징을 갖는 소형장치에서 실행 가능한 어플리케이션을 작성하기 위한 스펙과 기술의 집합이다[5, 9]. MIDlet은 J2ME에 정의된 MIDP(Mobile Information Device Profile) 호환 장치에서 실행 가능한 어플리케이션으로 MIDlet은 J2ME 스펙을 준수하는 장치에는 어디든지 동작할 수 있다. 모바일 어플리케이션은 다양한 장치에 설치되어 시스템과 사용자의 인터페이스 역할을 한다[21]. Myers[3]의 연구에 따르면 사용자 인터페이스(User Interface : UI)개발은 전체 프로그램 소스 코드 중 약 48%정도를 차지하며, 설계 단계의 45%, 구현 단계의 50% 정도의 시간을 소비한다.

현재 UI개발 방법으로는 개발도구, 프로그램의 운용환경에 따라 하위레벨 프로그래밍, 상위레벨 프로그래밍, 툴 킷의 사용, 비주얼 프로그래밍 등 다양한 방법이 사용되고 있다. 클라이언트-서버 환경에서 UI는 클라이언트 어플리케이션의 주요 기능에 해당하며, 클라이언트 시스템의 운영환경, 운영체제, 개발도구, 적용 장치 등의 요인들에 의해 영향을 받는다. 따라서 다양한 클라이언트 장치를 지원하기 위해서는 중복적인 클라이언트 개발이 발생할 수 있으며, 이는 시간과 비용의 낭비를 초래한다[12]. 이와 같은 문제점을 해결하기 위해 UI를 XML을 사용하여 명세하기 위한 다양한 UI 명세언어(User Interface Description Language : UIDL)가 개발되었다[14].

UIDL을 이용한 UI 개발은 개발도구나, 주변 환경에 독립

† 준 회 원 : 전남대학교 대학원 전산학과(이학박사)
†† 준 회 원 : 전남대학교 대학원 전산학과(이학박사)
††† 종신회원 : 전남대학교 전자컴퓨터공학부 교수
논문접수 : 2007년 11월 22일
수 정 일 : 1차 2008년 1월 4일, 2차 2008년 1월 16일
심사완료 : 2008년 2월 5일

적인 UI명세가 가능하고 자동화된 시스템을 통해 목표 플랫폼의 UI를 얻을 수 있는 장점이 있다. 대표적인 UIDL로는 UIML(User Interface Markup Language)[13], AUIML(Abstract User Interface Markup Language)[1], XIML(eXtensible Interface Markup Language)[2], USIXML(USer Interface eXtensible Markup Language)[20]등이 있다. 이러한 UIDL들은 각각 고유의 XML 어휘집(vocabulary)을 제공하고, 자동화된 도구를 통해 목표 플랫폼(target platform)의 UI를 생성하는데 이용된다. 하지만 기존 UIDL은 웹과 데스크톱 어플리케이션의 UI명세를 지원하며, 모바일 어플리케이션을 지원하지 않는다. 대신 모바일 장치에서 사용가능한 웹 문서인 WML(Wireless Markup Language)을 지원하는데, 이는 어플리케이션에 비해 상대적으로 그 응용범위가 제한적이다. 따라서 모바일 어플리케이션을 위한 UIDL이 필요하다.

본 논문에서는 자바의 모바일 플랫폼인 J2ME에 정의된 MIDlet의 UI명세를 위한 MIML(Midlet Interface Markup Language)을 제안한다. 또한, MIML로부터 MIDlet의 UI 생성과 관련된 자바 코드를 자동으로 생성하기 위한 규칙을 제안하고, 이 규칙을 이용하여 MIML로 작성된 UI를 자동으로 자바코드로 만들어주는 J2MERenderer를 구현하였다. 제안한 방법을 통해 다음과 같은 효과를 얻을 수 있다. 첫째, XML 기반의 언어를 사용함으로써 특정 개발 도구에 독립적인 MIDlet UI 명세가 가능하다. 둘째, 복잡한 소스코드 대신 직관적인 XML문서를 사용하여 UI를 구성함으로써 자바 언어에 미숙한 사용자도 MIDlet UI를 개발할 수 있다. 셋째, J2MERenderer를 제공함으로써 UI개발에 대해 RAD(Rapid Application Development)개발이 가능하다. 넷째, 하나의 MIML 명세로부터 유사한 UI를 갖는 어플리케이션을 빠르게 생성할 수 있으므로 재사용성을 증가시킬 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 UIDL과 UI생성에 관한 관련연구를 살펴보고, 3장에서는 모바일 UI 명세 언어인 MIML을 제안하고, MIML로부터 자바 코드를 생성하는 과정을 설명한다. 4장에서는 자동화 도구인 J2MERenderer의 설계와 구현에 대해 설명하고, 기존의 UIDL과 MIML을 비교하며, 마지막으로 5장에서는 결론 및 향후 연구방향을 제시한다.

2. 관련연구

본 장에서는 기존에 연구된 주요 UIDL과 UI 생성 기술에 대해 살펴본다.

2.1 사용자 인터페이스 명세 언어

UI 개발은 도구나 환경에 따라 다양한 방법이 존재한다. Visual Basic이나 C++과 같이 윈도우 시스템에서 제공하는 API(Application Programming Interface)를 사용하여 개발하는 방법, VB Script 나 Java Script와 같이 스크립트 언어를 사용하여 개발하는 방법, 두 개 이상의 개발 도구를 함께 사용하여 하이브리드 형태로 UI를 개발하는 방법 등이 있다. XML 기술이 발전하면서 특정 도구나 프로그래밍 언어에 종속적이지 않은 독립적인 UI 명세를 위한 다양한 UIDL이 등장하였다.

UIML은 가장 대표적인 UIDL로 Java AWT(Abstract Windows Toolkit), Swing, Visual Basic, VoiceXML[22], WML, HTML 등과 같은 다양한 형태의 UI를 표현할 수 있다. UIML은 3개의 섹션 즉, UI 명세 부분, 외부 엔티티와 연결할 수 있는 peers, 재사용을 위한 템플릿으로 구성된다[18]. 현재 UIML은 버전 3.0이 발표되었으며, 자바 언어의 데스크 톱 응용의 라이브러리인 AWT, Swing과 모바일 웹을 위한 WML 그리고 웹을 위한 HTML, VoiceXML등을 명세할 수 있는 어휘집이 개발되었다. 지원 도구로는 UIML 문서 편집과 Java AWT, Swing, HTML, WML의 UI를 생성할 수 있는 LiquidUI[19]가 있으며, 마이크로소프트사의 닷넷 플랫폼을 위한 UIML.NET[10] 과 Python 언어를 위한 변환기[6]도 연구되고 있다.

AUIML은 1998년 IBM사에서 장치 독립적인 마크업 언어를 개발하기 위한 DRUID 프로젝트로 개발된 언어이다. AUIML은 대부분 IBM사의 내부에서 사용할 목적으로 개발되었으며, 관련 도구로는 자바 통합개발환경인 Eclipse에서 사용할 수 있는 Visual Builder와 HTML 변환기를 제공한다[15].

XIML은 UI의 추상적인 모습과 구체적인 모습을 표현할 수 있는 언어로 Task, Domain, User, Presentation, Dialog 등 5가지의 컴포넌트로 구성된다. XIML도 UIML과 같이 다양한 종류의 클라이언트 장치를 위한 하나의 UI 정의를 지원한다. 하지만 XIML을 지원하는 도구는 XIML 코드 에디터 정도로 UI를 생성할 수 있는 렌더링 엔진을 가지고 있지 않다[25].

USIXML은 모델 기반의 UIDL로, taskModel, domainModel, mappingModel, contextModel, uiModel등 총 9개 모델을 사용하여 추상 UI 모델에서 최종적인 목표 플랫폼의 UI를 생성하기 위한 변환명세가 가능하다. 특히 USIXML은 MDA (Model Driven Architecture)기반 변환 타입(transformation type)을 지원한다. 지원 도구로는 HTML 페이지로부터 USIXML의 presentation model을 생성할 수 있는 ReversiXML과 USIXML 문서를 편집할 수 있는 GrafiXML등 다수의 지원도구를 제공한다. 정형화된 모델과 확장 MVP(Extended Model View Presenter) 모델을 적용한 XUIML[27]은 텍스트 편집기와 그래픽 편집기를 제공하며 Java와 C#언어로 변환이 가능하지만, 지원 플랫폼이 정의되어 있지 않다.

2.2 사용자 인터페이스 생성

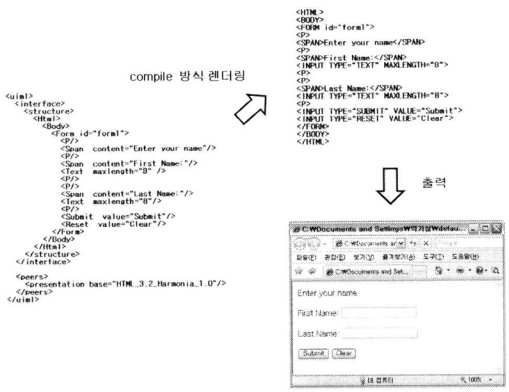
개발자가 UI를 개발하기 위해서는 UI를 특정 UIDL로 명세한 후, UIDL을 목표 플랫폼의 UI로 변환한다. 이때 UIDL 문서를 목표 플랫폼의 UI로 변환해주는 작업을 렌더링(Rendering)이라 하고 이때 사용하는 시스템을 렌더러(Renderer)라 한다.

렌더링은 크게 인터프리트 방식과 컴파일 방식이 있다. 인터프리트 방식은 UIDL문서를 파싱하여 목표 플랫폼의 UI를 바로 생성하는 방식이고, 컴파일 방식은 UIDL문서로부터 해당 프로그래밍 언어나 마크업 언어로 변환하는 방식이다[18]. 인터프리트 방식은 (그림 1)과 같이 UIML문서로부

(그림 1) 인터프리터 방식의 렌더링(Java AWT)

터 AWT기반의 UI를 얻을 수 있다. 그러나 이 방식은 소스코드는 생성하지 않고 결과 화면만을 출력하기 때문에 명세의 결과만을 확인할 수 있다. 컴파일 방식은 (그림 2)와 같이 UIML문서로 부터 HTML 문서를 생성하는 방식이다. 생성된 HTML문서는 추후 개발자가 다양한 코드를 추가하거나 수정되어질 수 있다. 이는 UI 생성의 융통성을 부여하는 장점이 있다. 기존의 변환기에서는 HTML이나 WML과 같은 마크업 언어에는 컴파일 방식을 사용하여 UI를 생성하고, 자바의 AWT나 Swing UI는 인터프리트 방식을 사용하여 UI를 생성함으로써 UI에 관한 코드를 얻을 수 없다[19]. 이는 명세와 개발이 독립된 작업으로 진행될 수 있는 문제점이 발생할 수 있으며, 결과적으로 명세의 효율성을 떨어뜨린다. 따라서 프로그래밍 언어에도 컴파일 방식을 적용한 렌더링을 적용함으로써 명세의 활용성을 높일 필요가 있다.

Jiancheng[24]은 UI 자동생성을 위해 상호작용 모델과 표현 모델을 정의하고, 이 두 모델과 UI 템플릿을 이용한 코드 생성방안을 제안하였다. Forstner[4]는 MDA와 도메인 명세언어(Domain Specific Modeling Language)기반의 VMTS (Visual Modeling and Transformation System)를 개발하여, 임베디드 모바일 플랫폼인 닷넷 Compact Framework와 Simbian Framework에 적용 가능한 C#코드와 C++코드를 생성하는 방안을 제시하였지만, 두 플랫폼의 UI 를 동시에 만족하는 명세는 불가능한 상태이다. Furtato[7]는 지식 기반 시스템(Knowledge Based System)과 USIXML을 이용한 인터페이스 생성을 위해 IKnowU framework를 개발하였다. 모바일 인터페이스를 생성하기 위한 SUPPLE[11]은 렌더러가 부재하고 J2ME를 지원하지 않는다. Concur Task Tree 모델 기반의 UI 생성도구인 TERESA[8]는 사용법이 복잡하고 모바일 어플리케이션 UI는 지원하지 않는다. 기존의 연구에서는 이론적인 방법들만 제시되어 실제적인 코드 생성기를 구현한 사례가 거의 없다. 또한 본 논문의 목표 플랫폼인 J2ME



(그림 2) 컴파일 방식의 렌더링(HTML)

를 위한 UI 명세를 지원하지 않는다. 따라서 본 연구에서는 대표적인 모바일 어플리케이션 모델인 J2ME의 MIDlet UI 명세를 위해 MIML과 코드 생성규칙을 제시하고, 활용 가능성을 보이기 위한 응용 시스템을 개발하였다.

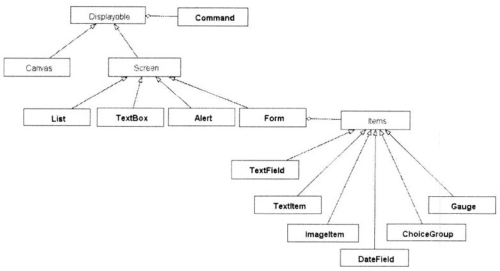
3. MIML(Midlet Interface Markup Language)

본 장에서는 J2ME의 응용 모델인 MIDlet의 UI 명세를 위해 XML 기반의 언어인 MIML을 제안한다. 그리고 제안된 MIML을 이용하여 작성된 UI명세를 목표 플랫폼인 J2ME의 MIDlet 코드로 변환해주는 소스코드 생성 규칙을 설명한다.

3.1 MIML의 구성요소

B'Far[17]는 모바일 컴퓨팅의 특징을 위치인식, 네트워크 QOS, 제한된 하드웨어 성능, 한정된 전원장치, 다양한 플랫폼등 8가지 요소로 정의하였다. 이러한 특징은 데스크 톱 어플리케이션이나 웹 어플리케이션과 구분되는 특징으로, 모바일 어플리케이션을 작성하는데 필요한 API를 구성하는데 영향을 미친다. 예를 들면, MIDlet의 UI와 이벤트에 관련된 클래스와 인터페이스의 수는 자바 어플리케이션의 경우에 비해 훨씬 적고 구조가 단순한 편이다. 또한 부동소수점을 지원하지 않으며 가비지 컬렉션을 위한 finalize()메소드를 지원하지 않는다. 이밖에도 단순한 예외 처리 구조와 자바 Applet과 유사한 라이프 사이클을 갖는다. 따라서, 자바 어플리케이션에 존재하는 main()메서드는 존재하지 않고, 프로그램이 미리 정의된 active, paused, destroyed 상태에 따라 동작하는 특징을 갖는다.

MIDlet UI를 명세하기 위해, 우선 J2ME의 UI와 관련된 클래스를 분석한다. J2ME에는 모바일 프로그램에 필요한 클래스와 인터페이스가 포함되어 있는데, MIML의 명세대상이 되는 클래스는 J2ME의 javax.microedition.lcdui 패키지 내의 UI 관련 클래스 정보가 해당된다. MIDlet의 화면 표시를 위해서는 lcdui 패키지내의 Displayable클래스로부터 상속받은 Canvas나 Screen 클래스를 이용하여 화면을 구성한



(그림 3) MIDlet UI 관련 클래스 관계

다. Canvas를 이용하는 방식은 점, 선, 면 등의 Canvas의 하위 클래스를 이용하는 방식으로 하위레벨(low level) UI 구성방식이고, Screen 클래스를 이용하는 방식은 컴포넌트 방식으로 미리 정의된 클래스의 객체를 이용하여 화면을 구성하는 방식으로 상위레벨(high level) UI 구성방식이다[16]. lcdui 패키지내의 UI 관련 클래스의 관계는 (그림 3)과 같이 표현할 수 있다. (그림 3)의 클래스 중 MIML의 명세대상이 되는 클래스는 굵은 글씨체로 표시된 클래스(Command, List, TextBox, Alert, Form, TextField, TextItem, ImageItem, DateField, ChoiceGroup, Gauge) 이다.

Screen 클래스를 이용하여 화면을 구성할 때에는 Screen 클래스의 하위 클래스인 TextBox, List, Alert, Form중 하나를 이용할 수 있다. TextBox, List, Alert은 독립적인 하나의 컴포넌트로 구성된 화면을 작성할 때 사용하고, Form을 이용할 경우, ChoiceGroup, DateField, TextField등 Item 클래스의 하위 클래스를 추가하여 화면을 작성할 수 있다. 따라서 상위레벨 UI를 이용하는 경우, 표현될 수 있는 화면은 크게 4가지 타입으로 구분된다. 위의 11가지 클래스의 UI와 관련된 속성을 추출하여 <표 1>과 같이 J2ME 어휘집을 정의하였다.

J2ME의 UI와 관련된 클래스 분석을 통하여 MIDlet의 UI명세 요소(element)를 10가지로 구분하였다. MIDlet의 UI관련 클래스는 MIML에서 하나의 UI 부분(<part>)에 해당하는데, 각 UI 부분은 해당 <part>의 class 속성에 따라 미리 정의된 속성(<property>)들을 포함하는 하나의 스타일(<style>)을 갖으며, 이러한 UI 부분이 모여서, 하나의 구조(<structure>)를 갖는다. UI의 이벤트(<event>)는 여러 개의 명령(<command>)으로 구성되며, 구조와 이벤트를 갖는 하나의 인터페이스(<interface>)를 정의한다. 인터페이스와 목표 플랫폼(<presentation>, <peers>)을 묶어서 하나의 MIML(<miml>)문서로 정의한다.

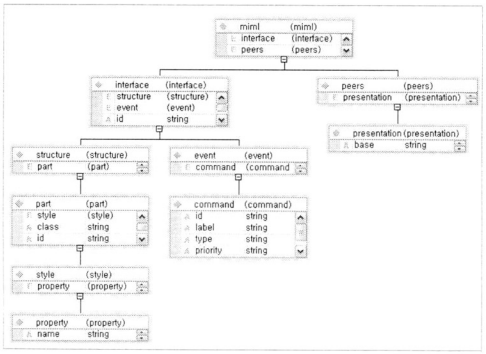
MIDlet의 UI 명세를 위한 MIML의 스키마는 (그림 4)와 같다. MIML문서는 최상위 요소인 <miml>로부터 시작하는

〈표 1〉 J2ME Vocabulary

Class	Property	Value
Alert	image	이미지 파일경로 혹은 URL
	text	문자열
	timeout	정수형
	title	문자열
	type	ALARM | CONFIRMATION | ERROR | INFO | WARNING
ChoiceGroup	elements	문자열 배열
	label	문자열
	type	EXCLUSIVE | IMPLICIT | MULTIPLE
DateField	label	문자열
	mode	DATE | DATE_TIME | TIME
Form	-	-
Gauge	initial value	정수형
	interactive	TRUE | FALSE
	label	문자열
	max value	정수형
ImageItem	image	이미지 파일 경로 혹은 URL
List	elements	문자열 배열
	title	문자열
	type	EXCLUSIVE | IMPLICIT | MULTIPLE
StringItem	mode	PLAIN | HYPERLINK | BUTTON
	label	문자열
	text	문자열
TextBox	constraints	ANY | CONSTRAINT_MASK | EMAILADDR | NUMERIC | PASSWORD | PHONENUMBER | URL
	maxsize	정수형
	title	문자열
	text	문자열
TextField	constraints	ANY | CONSTRAINT_MASK | EMAILADDR | NUMERIC | PASSWORD | PHONENUMBER | URL
	label	문자열
	max size	정수형
	text	문자열
Ticker	text	문자열
Command	label	문자열
	type	BACK | CANCEL | EXIT | HELP | ITEM | OK | SCREEN | STOP
	priority	정수형

〈표 2〉 MIML 요소와 기능

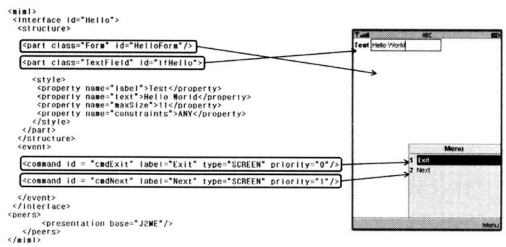
요소명	설명
miml	Root 요소
interface	인터페이스 구성요소를 포함하는 요소
structure	인터페이스의 구조 결정
part	인터페이스의 구성요소
style	인터페이스 구성요소의 스타일 결정
property	인터페이스 구성요소의 속성
event	이벤트의 구조 결정
command	커맨드 구성요소의 속성
peers	플랫폼 확장을 위한 요소
presentation	목표 플랫폼 설정



(그림 4) MIML 스키마 구조

계층 구조를 갖는다. UI를 구성하는 MIDlet의 UI클래스는 하나의 <part>에 대응되는데, <part>는 lcdui 패키지 내의 클래스에 대응하는 class 속성과, 객체 명에 해당하는 id 속성을 갖는다. 이 정보는 향후, 자바 코드 생성 시 클래스의 타입을 결정하는데 사용된다. 클래스의 타입이 결정되면, 해당 클래스의 여러 <property>를 지정할 수 있는데, 이는 앞서 제시된 <표 1>의 속성과 값을 사용한다. 예를 들어, 현재 클래스의 타입이 "StringItem"이라면 명세 가능한 <property>는 mode, label, text이며, 이 때 각 속성 값은 <표 1>에 따라 명세할 수 있다. mode의 경우 PLAIN, HYPERLINK, BUTTON 중 하나의 값을 가지며, label과 text 속성은 문자열 값을 가질 수 있다. 따라서 <property>의 집합으로 구성된 <style>은 해당 <part>의 UI 형태를 결정하게 된다. <표 1>의 클래스 중 Command 클래스는 MIDlet에서 메뉴를 구성하는 클래스로 MIML의 <command>요소에 명세 된다. Command 클래스는 코드 생성 시, 이벤트 관련 코드를 생성하는데 이는 UI를 구성하는 기타 클래스와 다른 독립된 규칙을 적용받게 된다. 따라서 별도의 요소로 구분되어 명세 된다. <peers> 요소는 UIML에서의 <peers>와 유사한 목적으로 MIML의 목표 플랫폼 확장을 위해 존재한다. 현재 지원하는 플랫폼은 J2ME이므로 현재의 명세는 J2ME로 한정한다.

앞서 제시된 어휘집과 스키마 구조를 갖는 완전한 MIML 문서의 예는 (그림 5)와 같다. 예에서는 Form과 TextField



(그림 5) MIML명세와 MIDlet 화면의 관계

그리고 2개의 Command로 구성된 UI를 표현한 것으로 UI 명세와 UI의 대응관계는 화살표로 표시하였다.

3.2 MIML을 MIDlet 코드로의 변환 규칙

MIML문서로 UI 명세를 한 후 목표 플랫폼에 사용될 소스코드를 자동으로 생성하는 것은 모바일 어플리케이션을 개발하는데 효율적으로 이용할 수 있다. MIML명세로 부터 MIDlet UI 생성에 필요한 자바 코드를 작성하기 위해 다음과 같은 8개의 규칙을 정의한다.

<규칙 1> MIDlet 구성요소를 위한 패키지를 생성할 소스 코드에 추가하고, MIDlet 클래스로부터 상속받은 자식 클래스를 정의한다. 이때, MIDlet 클래스의 이름은 MIML의 interface 요소의 id 속성 값으로 한다. 만약 MIML 문서 내에 하나 이상의 event 요소가 존재한다면 클래스 선언부에 CommandListener 인터페이스를 구현하는 부분이 추가된다.

MIML Form	<interface id="Hello">

Java Code	import javax.microedition.midlet.*;
	import javax.microedition.lcdui.*;
	
	public class Hello extends MIDlet(optional : implements CommandListener)
	{
	}

<규칙 2> MIDlet의 생명주기와 연관된 메소드(startApp(), pauseApp(), destroyApp())를 추가한다.

```
public void startApp() { }
public void pauseApp() { }
public void destroyApp(boolean unconditional){ }
```

<규칙 3> MIML의 <structure> 요소 내의 구성요소인 <part>는 lcdui 패키지내의 UI 객체로 선언된다.(이 때 <part> 요소의 class값과 id 값을 이용하여 해당 객체를 선언한다.)

MIML Form	<structure> <part class="Form" id="helloForm"> </structure>
Java Code	Form helloForm;

<규칙 4> MIML의 <structure> 요소내의 각 <part>는 해당 객체를 반환하는 get_객체명() 형식을 갖는 메소드를 생성한다.

MIML Form	<structure> <part class="Form" id="helloForm"> </structure>

Java Code	private Form get_helloForm()
	{
	}

<규칙 5> <property> 요소의 name 속성 값에 따라 <규칙 4>에서 정의된 메소드 내부에 UI 객체의 생성자 코드와 객체 반환 코드를 추가한다.

MIML Form	<style> <property name="label">TextField </property> <property name="text">TextFieldTest </property> <property name="maxSize">14</property> <property name="constraint">ANY </property> </style>

Java Code	if (TextField1 == null)
	{
	TextField1 = new TextField("Text Field", "TextField Test", 14,TextField.ANY);
	}
	
	return TextField1;

UI 객체의 생성자를 구현할 때, 별도의 객체가 필요한 경우에는 초기화에 필요한 객체의 생성자를 추가한다. 여기에 해당하는 UI 객체는 ImageItem의 Image 객체, List객체의 String[]객체가 있다.

MIML Form	<part class="ImageItem" id="imageItem1"> <style> <property name="image">log.jpg</property> </style> </part>

Java Code	private Image get_Image1()
	{
	if(image1 == null)
	{
	try
	{
	image1 = Image.createImage("./log.jpg");
	}
	catch(java.io.IOException exception)
	{
	}
	}
	return image1;
	}
	
	private ImageItem get_imageItem1()
	{
	if(imageItem1 == null)
	{
	imageItem1 = new ImageItem(null, get_Image1(), ImageItem.LAYOUT_DEFAULT, null);
	}
	return imageItem1;
	}

<규칙 6> 레이아웃은 <structure> 요소내의 <part>순서대로 화면의 위쪽부터 아래쪽으로 컨트롤을 배치한다. 즉 <규칙 3>, <규칙 4>, <규칙 5>에서 생성된 UI객체를 하나의 Form객체에 포함시킨다. Form 객체의 생성자의 매개변수로 <규칙 4>에서 생성된 get_객체명() 계열의 함수를 순서대로 추가되며 이는 UI의 레이아웃을 결정한다.

```
private Form get_helloForm()
{
 if(helloForm == null)
 {
  helloForm = new Form(null, new Item[] {
                                get_stringItem1(), get_imageItem1(),
                                get_stringItem2(), get_stringItem3(),
                                get_stringItem4()});
 }
 return helloForm;
}
```

<규칙 7> MIDlet의 화면을 나타내는 Display 객체에 현재 화면에 해당하는 Form 객체를 기본 화면으로 지정하는 initialize()메소드를 추가하고 이 메소드를 다시 startApp() 메소드에서 호출한다. 이때, 화면을 표현하는 객체가 Form 이외의 List, TextBox, Alert일 경우 해당되는 get 계열의 메소드를 작성하고 이를 Display.getDefaultDispay(this).setCurrent()메소드의 매개변수에서 호출한다.

```
public void startApp() {initialize();}

private void initialize()
{
 Display.getDefaultDisplay(this).setCurrent(get_helloForm());
}
```

<규칙 8> MIDlet의 이벤트 관련 코드생성을 위해 <event> 내의 <command>요소를 이용하여 관련코드를 작성한다. 관련코드로는 Command 객체 선언 및 생성, Form 객체 생성 시 command 객체 포함, 그리고 Form에 CommandListener 객체를 연결하는 코드가 포함된다.

MIML Form	<event> <command id = "cmdExit" label="Exit" type="SCREEN" priority="0"/> <command id = "cmdNext" label="Next" type="SCREEN" priority="1"/> </event>

Java Code	...
	Command cmdExit=new Command("Exit", Command.SCREEN, 0);
	Command cmdNext=new Command("Next", Command.SCREEN, 1);
	...
	private Form get_HelloForm()
	{
	...
	HelloForm.addCommand(cmdExit);
	HelloForm.addCommand(cmdNext);
	HelloForm.setCommandListener(this);
	return HelloForm;
	}
	...
	public void commandAction(Command c, Displayable s)
	{
	if (c == cmdExit)
	{
	}
	else if (c == cmdNext)
	{
	}
	}

〈표 3〉 생성된 자바 코드와 적용된 규칙

Java 코드	규칙
import javax.microedition.midlet.*; import javax.microedition.lcdui.*; public class Hello extends MIDlet implements CommandListener {	1
Form HelloForm; TextField tfHello;	3
Command cmdExit = new Command("Exit", Command.SCREEN, 0); Command cmdNext = new Command("Next", Command.SCREEN, 1);	8
public void startApp() { initialize(); } public void pauseApp() { } public void destroyApp(boolean unconditional) { }	2

private void initialize()
{
Display.getDisplay(this).setCurrent(get_HelloForm());
}
	7

private Form get_HelloForm() {	4

if (HelloForm == null)
{
HelloForm = new Form(null, new Item[] { get_tfHello() });
}
	6

HelloForm.addCommand(cmdExit); HelloForm.addCommand(cmdNext); HelloForm.setCommandListener(this);	8
return HelloForm; }	4
private TextField get_tfHello() {	4

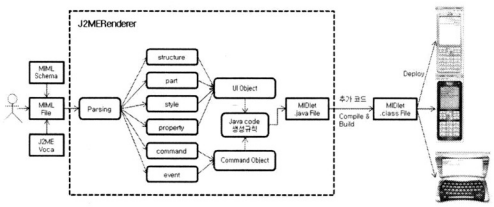
if (tfHello == null)
{
tfHello = new TextField("Test", "Hello World", 11, TextField.ANY);
}
return tfHello;
	5

}	4

public void commandAction(Command c, Displayable s)
{
if (c == cmdExit)
{
}
else if (c == cmdNext)
{
}
}
	8

}	1

MIML로 UI 명세를 한 후 목표 플랫폼에 사용될 소스코드를 자동으로 생성하는 위해서는 위에서 열거한 규칙을 다음과 같이 적용한다. <규칙 1>, <규칙 2>의 경우 기본적인 MIDlet를 구성하기 위한 기본 작업으로 각 1회 수행된다. <규칙 3>, <규칙 4>, <규칙 5>는 MIML문서의 <part> 수에 따라 반복적으로 수행될 수 있다. 이 단계들에서는 lcdui 패키지내의 UI관련 객체를 생성하고 각 객체의 속성 값을 <property> 명세에 따라 지정한다. <규칙 6>은 <규칙 3>, <규칙 4>, <규칙 5>에서 생성된 UI객체를 하나의 Form객체에 포함시키는 코드를 생성 한다. <규칙 7>에서는 Form 객체를 실제 표시 화면을 추상화한 Display객체를 지정한다. <규칙 8>에서는 Command객체를 <event> 구조에 따라 추가하고, 이벤트 처리 템플릿 코드를 생성한다. <표 3>은 앞서 명세 된 MIML(그림 5)부터 생성된 완전한 자바 코드와 적용된 규칙을 보여준다.



(그림 6) J2MERenderer 프로세스

4. J2MERenderer의 설계 및 구현

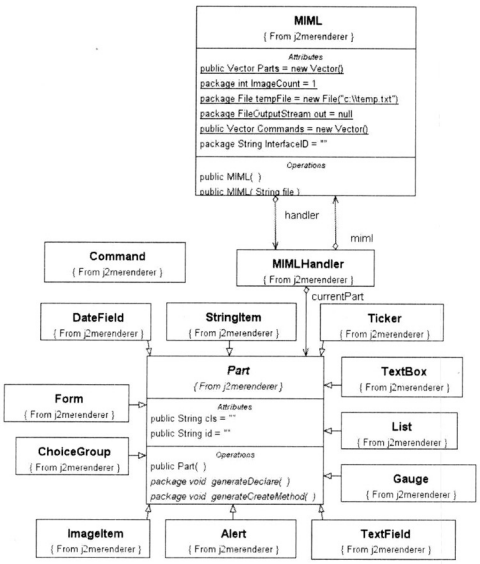
4.1 설 계

MIML를 이용하여 UI를 명세한 후 모바일 플랫폼에 사용할 UI를 생성하기 위해서는 변환기인 렌더러가 필요하다. 3장의 변환 규칙을 이용하여 MIML로부터 MIDlet의 UI 관련 자바 코드를 생성하는 시스템인 J2MERenderer를 개발하였다.

J2MERenderer의 동작과정은 (그림 6)과 같다. MIML 스키마와 MIML 어휘집을 사용하여 작성된 MIML문서는 J2MERenderer의 입력으로 사용된다. 입력된 MIML문서를 파싱하여, 요소 정보(structure, part, style, property, command, event)를 얻고, 이로부터 MIML에 명세된 UI 객체와 Command 객체를 추출한 후, 이를 3.2절에서 제안한 변환 규칙을 이용하여 자바 소스코드를 생성한다. 개발자는 생성된 코드를 기반으로 데이터 처리, 이벤트 처리, 통신 코드 등의 추가 기능을 구현하여 MIDlet 어플리케이션을 개발할 수 있다.

J2MERenderer 프로토타입의 클래스 구조는 (그림 7)과 같이 표현되는데, MIML문서로부터 생성된 객체는 (그림 7)에 표현된 Part 클래스의 하위클래스와 Command 클래스로 맵핑된다. J2MERenderer의 주요 클래스는 Part클래스, MIMLHandler 클래스, MIML클래스 등으로 구성되는데, Part 클래스는 javax.microedition.lcdui 패키지내의 UI 객체들을 대표하는 추상클래스로 generateDeclare() 메소드와 generateCreateMethod()를 선언한 추상클래스이다. 이로부터 상속 받은 Form, ImageItem, StringItem등의 클래스는 MIML문서에 표현된 각 UI 요소에 해당하는 속성을 가지며, Part 클래스의 추상메소드를 구현하였다. MIMLHandler 클래스는 SAX 파서로 프로토타입에서는 DefaultHandler 클래스로부터 상속받았다. MIML 클래스는 MIML문서를 추상화한 클래스로 입력으로 사용된 MIML문서로부터 파싱된 결과를 java.util.Vector 타입의 Parts 필드에 저장한다.

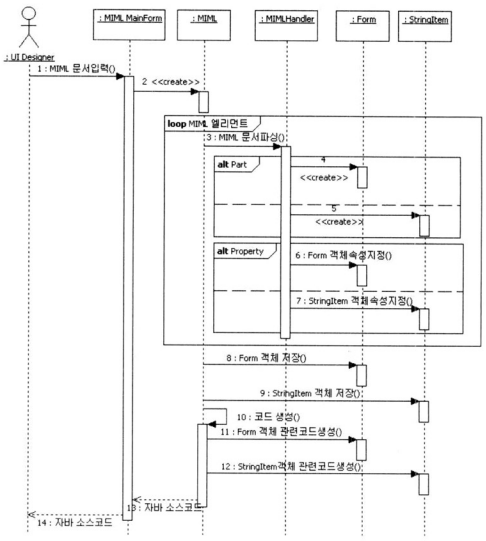
소스코드 생성 전체 과정은 MIML문서를 J2MERenderer의 MIML객체로 변환하는 과정과 MIML객체로부터 MIDlet 소스코드를 생성하는 과정으로 구성된다. 첫 번째 단계에서는 MIML 문서의 파싱을 통해 해당 UI 관련클래스와 필드의 값을 결정하여 그 값을 MIML객체의 Parts필드에 저장한다. 두 번째 단계는 MIML 객체의 Parts 필드로 부터 코드 생성규칙을 거쳐서 코드를 생성하는데 이때 UI 객체의 타입에 따라 해당 코드를 생성하기 위해 MIML 객체의 Parts 필드에 저장된 각 UI 객체의 타입을 식별하여 해당하는 generateDeclare() 메소드와 generateCreateMethod() 메소드를 수행하게 된다.



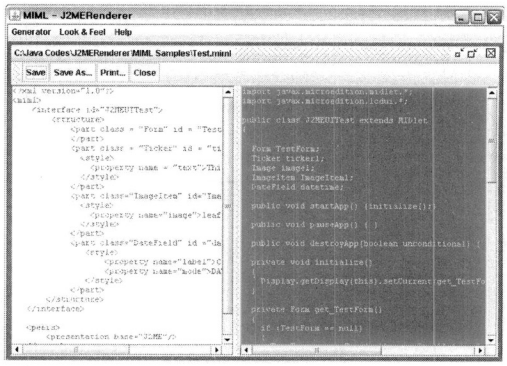
(그림 7) J2MERenderer의 클래스 구조

J2MERender에서 코드생성 과정을 순차다이어그램으로 표현하면 (그림 8)과 같다. User 액터가 MIML 문서를 J2MERenderer에 지정하면 MIMLMainForm에서는 User가 입력한 MIML 문서를 파싱하기 위해 MIML 객체를 생성한다. MIML 객체는 파싱을 위해 MIMLHandler를 사용해 MIML 문서내의 요소 값을 분석한다. 요소 값에 따라 해당 UI 객체를 생성하는데 예에서는 Form객체와 StringItem 객체를 각 하나씩 생성한다.

```
Input : MIML document
Output : Java source code

Algorithm Generate Java Code
  while (not end_of_file(miml)) do
    e ← miml.getElement()

    if e == "class" then
      create UI object obj of "class"

    else if e == "property" then
        p ← miml.getElement("property")
        obj.setProperty(p)
        Vector.add(obj)

    else if e == "command" then
      create Command object cmd
      cmd.set(id, label, type, priority)
      Vector.add(cmd)
    end if

  end while

  printDefaultMIDletCode();

  for each element ele of the Vector do
    if (ele instance of Part) then
      (UI Class)ele.printDeclare();
      (UI Class)ele.printMethod();

    else if (ele instance of Command) then
      printCommandDeclare();
      printUpdateMIDletDeclare();
      printUpdateFormMethod();
      printCommandAction();
    end if

  end for
```



(그림 8) J2ME Renderer 코드생성과정



(그림 9) J2MERenderer의 실행화면

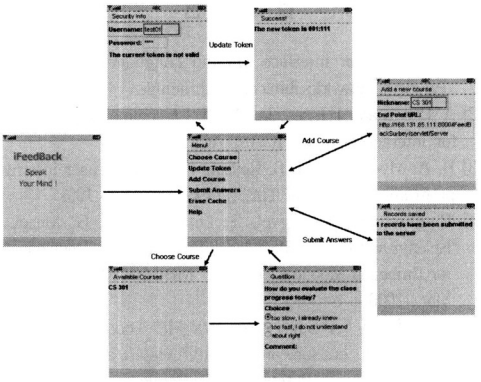
4.2 구 현

J2MERenderer는 JDK 1.6과 NetBeans 5.5를 사용하여 개발하였으며, MIDlet 실행환경은 Sun Java Wireless Toolkit 2.5 for CLDC를 이용하였다. J2MERenderer의 실행모습은 (그림 9)와 같다. 실행 창은 두 개의 영역으로 나뉘어져 있는데 파일 시스템에서 MIML 문서를 불러들이면 자동으로 생성된 소스코드를 화면에 출력한다. MIML 문서는 화면의 좌측 영역에 표시되고 생성된 소스 코드는 우측에 표시된다. 생성된 코드를 에뮬레이터에서 실행한 결과는 <표 4>와 같다. MIDlet UI의 바탕이 되는 Screen 클래스의 하위 클래스인 Form, Alert, TextBox, List로 구성된 화면을 변환 규칙에 따라 자바 코드를 생성하고, 생성된 화면은 J2MERenderer를 통해 생성된 코드를 컴파일, 빌드 과정을 거쳐 에뮬레이터에 출력한 결과이다.

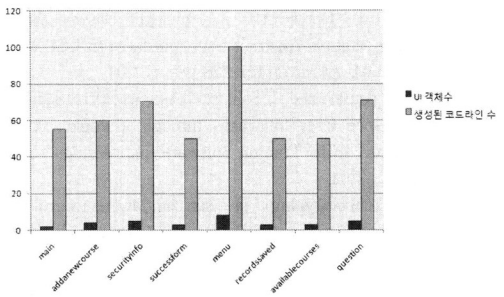
4.3 사례연구

J2MERenderer를 통해 생성되는 자바 소스 코드의 생산성을 실험하기 위해 Smart Client 프로그램의 일종인 iFeedBack[26] 응용프로그램의 UI생성에 적용하였다. iFeedBack은 교수자와 학습자 사이의 실시간 커뮤니케이션 채널을 제공하는 응용 프로그램이다. 사례에서는 iFeedBack 응용프로그램의 모든 UI를 제안한 MIML로 명세하고, 이를 개발한 J2MERenderer를 통해 UI를 작성하였다.

응용 프로그램의 시나리오에 따른 총 8개의 화면이 (그림 10)과 같이 정상적으로 생성되었다. 사례연구에 포함된 전체 UI 관련 객체 수는 33개이고, 이를 표현하기 위해 J2MERenderer를 통해 생성된 자바 코드 라인 수는 총 506라인이 생성되었다. 실제 iFeedBack 어플리케이션에는 쓰레드 처리, 데이터 교환등의 부가적인 코드가 추가되지만 여기서는 MIML을 이용하여 UI 생성에 관한 프로토타입 코드만을 생성한다. 각 화면별 필요한 객체수와 생성된 코드 라인 수는 (그림 11)와 같은 결과를 보였다.



(그림 10) MIML로부터 생성된 화면



(그림 11) 각 화면별 생성된 객체와 코드라인 수

4.4 평가

기존 UIDL과 제안한 MIML의 주요 특징을 정리하면 <표 5>와 같다. <표 5>에서 각 UIDL별 목표 플랫폼을 살펴보면, 기존 UIDL은 공통적으로 자바 언어를 위한 J2SE(Java Platform, Standard Edition)와 웹 그리고 모바일 웹을 지원하는 것을 알 수 있다. 그러나 J2ME와 같은 모바일 어플리케이션은 지원하지 않고 있다. 제안한 MIML은 기존의 UIDL이 제공하지 않은 J2ME의 UI 명세를 지원하며, J2MERenderer를 통해 명세로부터 코드를 생성할 수 있다. UIML, AUIML, USIXML은 UI생성을 위한 렌더러가 존재하지만 프로그래밍 언어와 목표 플랫폼을 완벽하게 지원하지는 못한다. MIML은 기존의 언어들에 비해 목표 플랫폼이 다양하지 못한 단점이 있지만, 이는 향후 WIPI나 닷넷 Compact Framework의 지원을 통해 해결 할 예정이다.

5. 결론 및 향후 연구과제

기존의 UIDL은 주로 웹과 데스크톱 어플리케이션의 UI 명세에 활용되고 있으나, 모바일 어플리케이션을 위한 명세는 지원하지 않는다. 이에 본 논문은 자바의 모바일 플랫폼인 J2ME의 UI 명세를 위해 스키마, 어휘집을 정의하고, MIML (Midlet Interface Markup Language)를 제안하였다. 제안한 MIML의 효율적 사용을 위해 MIML명세로 부터 UI 관련 자바 소스코드를 자동으로 생성하기 위한 규칙을 제시하였다. 이 규칙을 이용하여 MIML로부터 소스코드를 자동생성해 주는 자동화 도구인 J2MERenderer를 개발하였다. 사례연구 결과 제안한 MIML은 MIDlet의 상위레벨 UI 명세를 완벽히 지원하였다. MIML로 명세된 UI를 J2MERenderer를 통해 자바 코드를 생성함으로써 J2ME UI에 대한 기반 지식이 없어도 안전하고 빠르게 MIDlet UI에 관련된 자바 소스코드를 제공받을 수 있다. 향후, MIML이 단일 플랫폼을 지원하는 한계를 극복하고, 활용성을 극대화하기 위해 비슷한 사용자 UI를 갖는 WIPI(Wireless Internet Platform for Interoperability)[23]나 닷넷 모바일 플랫폼으로의 확장을 위한 연구가 필요하다.

〈표 5〉 주요 UIDL의 특징

UIDL	렌더러	Tag 수	지원언어	목표 플랫폼
UIML	△	36	Java(AWT, SWING), HTML, WML, VoiceXML, C++ 등	J2SE, Web, Mobile Web, PalmOS, Windows
AUIML	△	55	HTML, DHTML, Java Swing, WML	J2SE, Web, Mobile Web
XIML	X	33	Java, HTML, WML	J2SE, Web, Mobile Web
USIXML	△	약 350 (9개 모델)	Java, C++, HTML, WML 등	J2SE, Web, Mobile Web, PalmOS
MIML	O	10	Java	J2ME

참 고 문 헌

[1] Abstract User Interface Markup Language Toolkit, http://www.alphaworks.ibm.com/tech/auiml

[2] A. Puerta and J. Eisestein, "XIML : a common representation for interaction data," IUI'02, ACM, 2002.

[3] B. A. Myers and M. B. Rosson, "Survey on user interface programming," SIGCHI'92, pp.195-202, May, 1992.

[4] B. Forstner, L. Lengyel, T. Levendovszky, G. Mezei, I. Kelényi and H. Charaf, "Model-Based System Development for Embeded Mobile Platforms," MBD/MOMPES'06, IEEE, Mar., 2006.

[5] C. E. Ortiz, "A Survey of Java ME Today," Nov. 2007. http://developers.sun.com/mobility/getstart/articles/survey/

[6] E. Cherkashin, "Python UIML Renderer," 2001. http://freshmeat.net/projects/pyuiml

[7] E. Furtado, V. Furtado, K. S. Sousa, J. Vanderdonckt and Q. Limbourg, "KnowiXML:A Knowlegde-Based System Generating Multiple Abstract User Interfaces in USIXML," TAMODIA'04, ACM, pp.121-128, Nov., 2004.

[8] G. Mori, F. paternò and C. Santoro, "Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions," IEEE Tran on Software Engineering, Vol.30, No.8, Aug., 2004.

[9] Java ME Technology, http://java.sun.com/javame/technology/

[10] K. Luyten and K. Coninx, "UIML.Net: an Open Uiml Renderer for the .Net Framework," CADUI'2004, pp.260-273, 2004

[11] K. Gajos and D. S. Weld, "SUPPLE: Automatically Generating User Interfaces," IUI'04, ACM, Jan., 2004.

[12] M. Abrams, C. Phanouriou, A. Batongbacal, S. Williams and J. Shuster, "UIML: An Appliance Independent XML User Interface Language," In Proceedings of the Eighth International World Wide Web Conference, pp.617-630, 1999.

[13] M. Abrams and C. Phanouriou, "UIML: An XML Language for Building Device-Independent User Interface," XML'99, 1999.

[14] N. Souchon and J. Vanderdonckt, "A Review of XML-compliant User Interface Description Languages," DSV-IS 2003, LNCS 2844, pp.377-391, 2003.

[15] P. Azevedo, R. Merrick and D. Roberts, "OVID to AUIML a user oriented interface modeling," TUPIS'00, Oct. 2000.

[16] Q. H. Mahmoud, "MIDP GUI Programming: Programming the Phone Interface," 2000. http://developers.sun.com/techtopics/mobility/midp/articles/ui/.

[17] R. B'Far, 'Mobile Computing Principles,' Cambridge University Press, 2005.

[18] UIML Specification Draft 3.1, http://www.oasis-open.org/committees/documents.php?wg_abbrev=uiml.

[19] UIML Tools, http://www.uiml.org/tools.

[20] USIXML Forum, http://www.usixml.org.

[21] V. Lee, H. Schneider and R. Schell, 'Mobile Applications Architecture, Design, and Development,' Prentice Hall PTR, 2004.

[22] VoiceXML Forum, http://www.voicexml.org.

[23] WIPI Forum, http://www.wipi.or.kr.

[24] W. Jiancheng, L. Xudong and L. Lei, "A Model of User Interface Design and Its Code Generation," IRI'07, IEEE, pp. 128-133, Aug., 2007.

[25] XIML Forum, http://www.ximl.org

[26] Y. Michael, 'Enterprise J2ME DEVELOPING MOBILE JAVA APPLICATION.' Prentice Hall PTR, 2004.

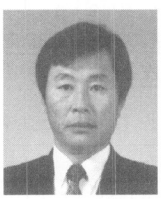
[27] 최종명, 신경희, 유재우. "사용자 인터페이스를 위한 MVP기반의 XML언어," 정보과학회 논문지, 제29권 제12호, pp.947-956, 2002.

박 기 창

e-mail : kcpark@chonnam.ac.kr
2001년 전남대학교 물리학과(이학사)
2003년 전남대학교 대학원 전산학과 (이학석사)
2005년 전남대학교 대학원 전산학과 수료 (이학박사)
관심분야 : CASE 도구, 소프트웨어 가시화, 정형기법

서 성 채

e-mail : scseo@chonnam.ac.kr
1994년 전남대학교 전산학과(이학사)
1997년 전남대학교 대학원 전산학과 (이학석사)
2006년 전남대학교 대학원 전산학과 (이학박사)
관심분야 : 소프트웨어 모델링, 요구공학, 소프트웨어 보안, 정형기법 등



김 병 기

e-mail : bgkim@chonnam.ac.kr
1978년 전남대학교 수학교육과(이학사)
1980년 전남대학교 대학원 수학과 (이학석사)
2000년 전북대학교 대학원 수학과 (이학박사)
1981년~현 재 전남대학교 전자컴퓨터공학부 교수
1995년~2006년 한국 정보처리학회 이사 및 부회장
2007년 한국정보처리학회장
관심분야 : 소프트웨어공학, 객체지향시스템, 컴포넌트 개발 등