사용자 인터페이스 명세 언어를 이용한 위피 비즈니스 프로그램 저작도구 구현

Implement of The Authoring Tool for WIPI Business Program using UIDL(User Interface Description Language)

이동수*, 박기창**, 김병기*
전남대학교 전자컴퓨터공학과*, 전남대학교 전산학과**

Dong-Su Lee(tnqnffn@naver.com)*, Ki-Chang Park(kcpark@dsu.ac.kr)**,
Byung-Ki Kim(bgkim@jnu.ac.kr)*

요약

현재 위피 응용프로그램을 개발하기 위한 저작도구는 코드 작성의 편의성이 부족하고, 시각화를 배제한 문법 중심의 프로그래밍 도구가 대부분이다. 이로 인해 위피 응용프로그램 개발 시 개발자가 전체 개발과정에 걸쳐서 직접 코딩하여 개발하고 있다. 본 논문에서는 위피 응용프로그램을 신속하고 용이하게 개발할 수 있는 절차로써 위피API의 상위레벨(High level) 컴포넌트를 그래픽 컨트롤로 시각화하고, 이를 이용하여 시스템에서 제공한 모바일 레이아웃에 개발자가 쉽게 디자인한 후, UI 명세 언어와 소스 코드로 생성 해주는 과정을 거치는 개발 방안을 제시한다. 이를 위하여 위피 Jlet의 UI 명세 언어인 JIML(Jlet Interface Markup Language)을 제안하고, 제안한 JIML로부터 위피 Jlet의 UI와 관련된 위피 코드로의 생성을 위한 변환 규칙을 제시한다. 또한 위의 개발 과정을 자동으로 처리하는 시스템을 구현하였다. 구현한 시스템은 위피 비즈니스 프로그램 개발 시 효율성을 제공하고, 빠른 개발이 가능하도록 도와준다.

■ 중심어 : | WIPI | UIDL | RAD | XML | Code Generator | MIML |

Abstract

At present, Authoring tools, which are used to develop the WIPI applications are less convenient to be coded. Usually, the programming tools only focus on the grammar while scarcity of visualization. Developer forced directly codes all times during the development by this cause. As a procedure for rapid WIPI Application Development, in this paper, WIPI API High Level Component is first visualized, in order words, graphic control is developed. Second this control is used in designing the mobile layout. Then UI Markup Language and source code can be generated automatically. With this view, we propose the JIML(Jlet Interface Markup Language) with UI Markup Language based XML of WIPI Jlet Platform and also present the transformable rule for generation to the WIPI code about WIPI Jlet UI from offered JIML. Also we implement the WIPI Jlet Generation System to make the generation to JIML, WIPI code. The Implemented System provides efficiency when developing WIPI Business Application, and helps to enable rapid development.

■ keyword : | WIPI | UIDL | RAD | XML | Code Generator | MIML |

접수번호 : #081215-004
접수일자 : 2008년 12월 15일
심사완료일 : 2009년 02월 23일
교신저자 : 김병기, e-mail : bgkim@jnu.ac.kr

I. 서 론

무선인터넷 콘텐츠의 경우 사용자의 요구사항이 점점 다양해지고 있기 때문에 시장 경쟁력을 가지기 위해서는 새로운 형태의 콘텐츠를 신속하게 저렴한 비용으로 개발할 수 있어야 한다[1].

현재 위피 응용프로그램을 저작하기 위한 도구는 텍스트 편집기 수준이 대부분이고, 문법 중심으로 설계되어 있어 전체 개발과정을 직접 코딩해야 하며, 인터페이스 또한 편의성이 배제되어 있다. 따라서 위피 응용프로그램을 신속히 개발하고 편의성 향상을 증대할 수 있는 저작 시스템에 대한 연구가 필요하다.

빠른 위피 응용프로그램 개발을 위한 절차로 본 논문에서는 위피 API의 상위레벨 컴포넌트를 GUI 위젯(Widget), 즉 그래픽 컨트롤로 시각화하고, 이를 이용하여 시스템에서 제공한 모바일 레이아웃에 개발자가 쉽게 디자인한 후, UI 명세 언어와 소스 코드로 생성 해주는 과정을 거치는 개발 방안을 제시한다. 이를 위하여 위피 Jlet 플랫폼의 XML 기반의 UI 명세 언어인 JIML(Jlet Interface Markup Language)을 제안하고, 제안한 JIML로부터 위피 Jlet의 UI와 관련된 위피 코드로의 생성을 위한 변환 규칙을 제시한다. 이 규칙을 이용하여 JIML을 위피 코드로 변환해주는 위피 응용프로그램 생성 시스템을 구현하였다.

시스템은 GUI에서 바로 위피 코드로 변환하는 인터프리터 방식인 아닌 컴파일 방식을 지향한다. 이는 컴파일 방식을 통하여 본 논문에서 제안한 위피 Jlet UI 명세언어인 JIML을 생성함으로써, 다음과 같은 효과를 얻을 수 있기 때문이다. 첫째, XML 기반의 언어를 사용함으로써 특정 개발 도구에 독립적인 Jlet UI 명세가 가능하다, 둘째, 시각적인 컨트롤 제공과 복잡한 소스코드 대신 프로그램 구조를 직관적으로 볼 수 있는 XML 문서 제공을 통하여 위피 언어에 미숙한 사용자도 위피 비즈니스 프로그램을 빠르게 개발할 수 있다. 셋째, 기존에 생성된 JIML를 이용하여 유사한 UI를 빠르게 개발할 수 있다. 넷째, 제안한 JIML 태그 숙지를 통하여 간단한 UI를 개발자가 직접 명세하여 소스코드로 변환할 수 있다. 다섯째, 기타 다른 UIML이나 다른 명세언어로 변환기만 있으면 변환이 용이하다.

II. 관련연구

위피 콘텐츠 개발 방법, 위피 API의 상위레벨 컴포넌트, 이동통신 3사의 에뮬에이터, GUI 위젯에 관한 분석은 기존의 위피 연구를 참고 하였다[2-4]. 이번 장에서는 모바일 인터페이스 명세를 위해서 UIDL(User Interface Description Language)과 UI 생성기술에 대해서 알아본다.

1. 기존의 범용 개발 도구 및 위피 연구동향

기존 위피 응용프로그램에 관련해서는 다양한 운영체제에서 사용가능한 CNU 위피 에뮬레이터[1], GVM-to-WIPI 플랫폼 컨버팅(Converting)[10] 등이 연구되었다. 하지만 위피 프로그램 저작 도구에 대한 연구는 진행되고 있지 않았다. 이는 위피 비즈니스 프로그램 개발보다 게임 개발의 수요가 많고, 게임 개발은 비즈니스 프로그램처럼 위피 상위레벨 API를 거의 활용을 하지 않고, 코드 개발이 정형화 되어있지 않으며, 키이벤트 및 이미지 배치 위주의 개발이기 때문이다.

[표 1]은 기존의 모바일 프로그래밍을 위한 도구를 분석하여 정리하였다.

표 1. 모바일 프로그래밍 도구 분석

기능 / 개발 도구	GUI 기반	마법사 기능	모바일 코드 개발의 편의성	모바일 UI 디자인 기능
이클립스	O	O	△	X
넷빈즈	O	O	△	X
GNEX	O	O	△	X
KIS	O	O	△	X

O : 기능 있음, △ : 부족함, × : 기능 없음

모바일 프로그래밍 도구는 개발을 위한 기본 기능들을 갖추고 있다. 이클립스, 넷빈즈 같은 개발 도구는 마법사 기능, 프로젝트 생성 및 빌드 등을 지원한다. 하지만 모바일 프로그램 코드 개발을 위한 도구로써 텍스트

편집기와 크게 차이가 나지 않는다. 이동통신 3사 중 KTF에서 개발한 KIS(KTF Integrated SDK)는 위피 응용프로그램 저작도구로써 리소스 관리, 에뮬레이터 연동, 소스 에디터 등의 기능을 지원하는 유일한 통합 도구이다. 하지만, KTF 위피만을 지원하므로 이동통신 3사 통합 개발 도구라고 하기에는 미흡하다. 또한 소스 편집기 같은 경우는 기본 라이프 사이클 정도의 위피 코드 생성은 가능하지만, 텍스트 편집기와 크게 차별성을 찾기 힘들다. GNEX 플랫폼 개발을 지원하는 IDE는 Mobile C로 개발되며, GNEX IDE, 에뮬레이터, 리소스 편집기 등의 기능을 지원한다. 이 또한 KIS 기능과 크게 다르지 않으며 텍스트 코드 작업이 주를 이룬다. 그리고 위피 플랫폼을 지원하지 않고 SKT GNEX 플랫폼 개발을 지원한다. 본 논문의 자동화시스템은 비쥬얼 컨트롤을 이용하여 위피 UI 관련 코드를 자동 생성 하므로 위피 비즈니스 프로그램을 쉽게 개발할 수 있다.

2. UIDL과 UI 생성기술

XML 기술이 발전하면서 플랫폼에 종속적이지 않는 UI 명세를 위한 다양한 UIDL이 등장하였다. 대표적인 UIDL로는 UIML(User Interface Markup Lagnuage)[6], AUIML(Abstract User Interface Markup Language)[7], XIML(eXtensible Interface Markup Language)[8], MIML(Midlet Interface Markup Language)[5] 등이 있다. UIML은 가장 대표적인 UIDL로 Java AWT, Swing, VB, WML 등과 같은 다양한 형태의 UI를 표현할 수 있다. AUIML은 1998년 IBM사에서 장치 독립적인 마크업 언어를 개발하기 위한 DRUID 프로젝트로 개발된 언어이다. XIML은 UI의 추상적인 모습과 구체적인 모습을 표현할 수 있는 언어이다. XIML도 UIML과 같이 다양한 종류의 클라이언트 장치를 위한 하나의 UI 정의를 지원한다. 이러한 UIDL들은 각각 고유의 XML 어휘집(vocabulary)를 제공하고, 자동화된 도구를 통해 목표 플랫폼의 UI를 생성하는데 이용된다[5].

표 2. 주요 UIDL 특징

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

UIDL의 특징은 [표 2]에 정리되었으며, 각 UIDL 별 목표 플랫폼을 살펴보면 자바 언어를 위한 J2SE, 웹 그리고 모바일 웹을 지원하는 것을 알 수 있다. 그러나 MIML[5]을 제외하고는 위피와 같은 모바일 응용프로그램은 지원하지 않고 있다. 최근에 연구된 MIML은 모바일 자바로 작성하는 플랫폼인 J2ME를 지원하지만, 위피 플랫폼은 지원하지 않고 있다. 따라서 위피 응용프로그램을 위한 위피 Jlet 사용자 인터페이스 XML언어의 확장이 필요하다.

UI 생성 기술에는 크게 인터프리트 방식과 컴파일 방식이 있다. 인터프리트 방식은 UIDL을 파싱하여 목표 플랫폼의 UI를 바로 생성하는 방식이고, 컴파일 방식은 UIDL문서로부터 해당 프로그래밍 언어나 마크업 언어로 변환하는 방식이다. [그림 1]은 두 방식의 차이를 나타낸다. 인터프리트 방식은 소스코드는 생성하지 않고 오직 결과 화면만을 출력하기 때문에 명세의 결과만을 확인할 수 있다. 컴파일 방식은 인터프리트 방식과는 달리 소스코드가 출력되고, 이를 추가하거나 수정할 수 있어서 UI 생성의 융통성을 부여하는 장점이 있다.

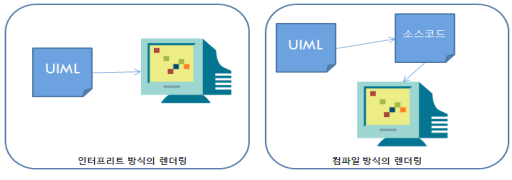


그림 1. 인터프리트 방식과 컴파일 방식의 차이

기존의 UIDL의 변환기에서는 HTML이나 WML과 같은 마크업 언어에는 컴파일 방식을 사용하여 UI를 생성하고, 자바의 AWT나 Swing UI는 인터프리트 방식

을 사용하여 UI를 생성함으로써 UI에 관한 코드를 얻을 수 없다. 이는 명세와 개발이 독립된 작업으로 진행될 수 있는 문제점이 발생할 수 있으며 결과적으로 명세의 효율성을 떨어뜨린다. 따라서 프로그래밍 언어에도 컴파일 방식을 적용한 자동생성기술을 적용함으로써 명세의 활용성을 높일 필요가 있다[5].

Ⅲ. JIML(Jlet Interface Markup Language)

위피 Jlet의 UI 명세를 위해 XML 기반의 언어인 JIML을 제안한다. 그리고 제안한 JIML을 이용하여 자동으로 생성된 UI 명세를 위피의 Jlet 코드로 변환해주는 소스코드 생성 규칙을 기술함으로써 다음 장에서 기술하는 위피 응용프로그램 생성 시스템을 구현할 수 있는 기초를 제공한다.

1. JIML 구성요소

JIML의 명세를 위해서 위피 API 상위레벨 클래스의 UI와 관련된 속성을 추출하여 [표 3]과 같이 Jlet 어휘집을 정의하였다.

표 3. Jlet Vocabulary

Class	Property	Value
ShellComponent	title	문자열
FormComponent	interactive	TRUE | FALSE
ButtonComponent	label	문자열
CheckboxComponent	text	문자열
	interactive	TRUE | FALSE
	group	문자열
ComboComponent	text	문자배열
DataFieldComponent	mode	MODE_TIME | MODE_DATE | MODE_TIME_DATE
ImageComponent	image	이미지 파일 경로 혹은 URL
ListItemComponent	label	문자열
	type	EXCLUSIVE | IMPLICIT | MULTIPLE
LabelComponent	label	문자열
ProgressComponent	initial value	정수형
	interactive	TRUE | FALSE
TextBoxComponent	constraints	CONSTRAINT_NUMBER | CONSTRAINT_PASSWORD | CONSTRAINT_ANY
	text	문자열
TextFieldComponent	constraints	CONSTRAINT_NUMBER | CONSTRAINT_PASSWORD | CONSTRAINT_ANY
	text	문자열
TickerComponent	text	문자열

위피 Jlet의 UI명세 element를 10가지로 구분하여 [표 4]에 정리하였다. Jlet의 UI관련 클래스는 JIML에서 하나의 UI 부분(<part>)에 해당하는데, 각 UI 부분은 해당 <part>의 클래스 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>)를 묶어서 하나의 JIML(<jiml>)문서로 정의한다.

표 4. JIML 요소와 기능

요소명	설명
jiml	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

Jlet의 UI 명세를 위한 JIML의 스키마는 [그림 2]와 같다.

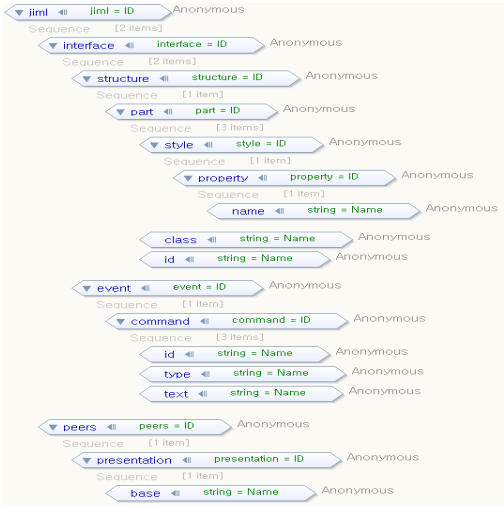


그림 2. JIML 스키마 구조

JIML문서는 최상위 요소인 <jiml>로부터 시작하는 계층 구조를 갖는다. UI를 구성하는 Jlet의 UI클래스는 하나의 <part>에 대응되는데, <part>는 lwc 패키지 내의 클래스에 대응하는 클래스 속성과, 객체 명에 해당하는 id 속성을 갖는다. 이 정보는 향후, 위피 코드 생성 시 클래스의 타입을 결정하는데 사용된다. 클래스의 타입이 결정되면, 해당 클래스의 여러 <property>를 지정할 수 있는데, 이는 앞서 제시된 [표 3]의 속성과 값을 사용한다. 예를 들어, 현재 클래스의 타입이 "TextFieldComponent"라면 명세 가능한 <property>는 constraints, text이며, 이 때 각 속성 값은 [표 3]에 따라 명세할 수 있다. constraints의 경우 CONSTRAINT_NUMBER, CONSTRAINT_PASSWORD, CONSTRAINT_ANY 중 하나의 값을 가지며, text 속성은 문자열 값을 가질 수 있다. 따라서 <property>의 집합으로 구성된 <style>은 해당 <part>의 UI 형태를 결정하게 된다. <peers> 요소는 UIML에서의 <peers>와 유사한 목적으로 기존에 제안한 MIML에 목표 플랫폼 확장 가능한 위피 UI XML 언어인 JIML을 위해 존재한다. 현재 지원하는 플랫폼은 위피 이므로 현재의 명세는 위피 Jlet으로 한정한다.

앞서 제시된 어휘집과 스키마 구조를 갖는 완전한 JIML 문서의 예는 [그림 3]과 같다.

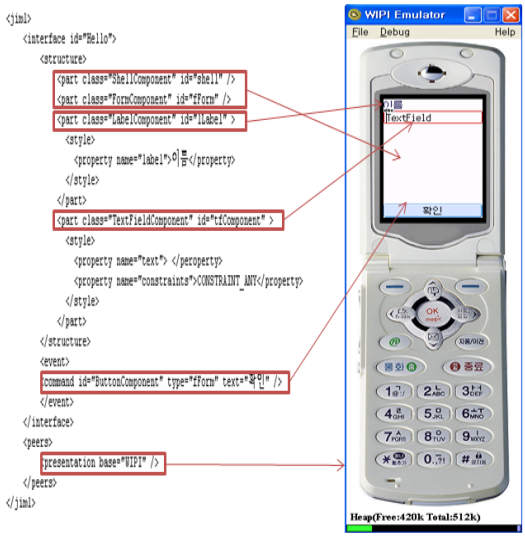


그림 3. JIML 명세의 예

2. 위피 Jlet 코드로의 변환 규칙

<규칙 1> Jlet 구성요소를 위한 패키지를 생성할 소스 코드에 추가하고, Jlet 클래스로부터 상속받은 자식 클래스를 정의한다. 여기서 Jlet 클래스의 이름은 JIML의 interface 요소의 id 속성 값으로 한다. 만약 JIML 문서 내에 하나 이상의 이벤트 요소가 존재한다면 클래스 선언부에 ActionListener 인터페이스를 구현하는 부분이 추가된다.

JIML	〈interface id="Hello"〉
위피 코드	import org.kwis.msp.lcdui.*; import org.kwis.msp.lwc.*; public class Hello extends Jlet(optional : implements ActionListener){}

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

<규칙 3> JIML의 <structure> 요소 내의 구성요소인 <part>는 lwc 패키지 내의 UI 컴포넌트 객체로 선언된다. 이 때 <part> 요소의 class와 id 값을 이용하여 해당 객체를 선언한다.

JIML	〈structure〉 〈part class="ShellComponent" id="shell"〉 〈/structure〉
위피 코드	ShellComponent shell;

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

JIML	〈structure〉 〈part class="ShellComponent" id="shell"〉 〈/structure〉
위피 코드	private ShellComponent get_shell0(){}

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

JIML	〈style〉 〈property name="text"〉TextField〈/property〉 〈property name="constraints"〉CONSTRAINT_ANY〈/property〉 〈/style〉

위피 코드	if(tfComponent0 == null)
	{
	tfComponent0 = new TextFieldComponent("TextField", TextComponent.CONSTRAINT_ANY);
	}
	return tfComponent0;

<규칙 6> Jlet의 경우 레이아웃은 <structure> 요소내의 <part> 순서대로 화면의 위쪽부터 아래쪽으로 컨트롤을 배치한다. 즉 <규칙 3>, <규칙 4>, <규칙 5>에서 생성된 UI 객체를 ShellComponent 또는 FormComponent 객체에 포함시킨다.

<규칙 7> Jlet의 화면을 나타내는 ShellComponent 객체를 기본 화면으로 지정하는 initialize()메소드를 추가하고, 이 메소드를 다시 startApp() 메소드에서 호출한다. initialize()메소드에는 shell에 담긴 화면을 나타내는 show()메소드가 정의된다.

```
private void initialize(){
    get_shell().show();
}
```

<규칙 8> Jlet의 이벤트 관련 코드생성을 위해 <event>내의 <command>요소를 이용하여 관련코드를 작성한다.

JIML	〈event〉 〈command id="ButtonComponent" type="fForm" text="확인" /〉 〈/event〉

위피 코드	ButtonComponent bButton = new ButtonComponent("확인", null);
	
	private void initialize(){
	bButton.setActionListener(this, bButton);
	}
	private ShellComponent get_shell(){
	shell.setCommand(bButton, false);
	}
	public void action(Component cmp, Object o){
	if(cmp == bButton){
	}
	}

JIML로 자동으로 명세된 후, 위피 코드를 자동으로 생성하기 위해서 위에서 열거한 8가지 규칙을 다음과 같이 적용한다. <규칙 1>, <규칙 2>의 경우 기본적인 Jlet을 구성하기 위한 기본 작업으로 각 1회 수행된다. <규칙 3>, <규칙 4>, <규칙 5>는 JIML문서의 <part> 수에 따라 반복적으로 수행될 수 있다. 이 단계들에서는 lcdui, lwc 패키지내의 객체를 생성하고, 각 객체의 속성 값을 <property> 명세에 따라 지정한다. <규칙 6>은 <규칙 3>, <규칙 4>, <규칙 5>에서 생성된 객체를 ShellComponent 또는 FormComponent 객체에 포함시키는 코드를 생성 한다. <규칙 7>에서는 화면을 생성시킬 객체를 지정한다. <규칙 8>에서는 이벤트 관련 객체를 처리하고 템플릿 코드를 생성한다. [표 5]는 JIML로부터 생성된 위피 코드와 적용된 규칙을 보여준다.

표 5. 생성된 위피 코드와 적용된 변환 규칙

위피 코드	규칙
import org.kwis.msp.lcdui.*; import org.kwis.msp.lwc.*; public class Hello extends Jlet implements ActionListener{	1
ShellComponent shell; FormComponent form; LabelComponent lLabel0; TextFieldComponent tfComponent0;	3
ButtonComponent bButton = new ButtonComponent("확인", null);	8

protected void startApp(String args[]) { initialize(); } protected void pauseApp(){} protected void resumeApp(){} protected void destroyApp(boolean b){}	2
private void initialize(){ get_shell().show();	7
bButton.setActionListener(this, bButton);	8
}	7
private ShellComponent get_shell(){	4

if(shell == null){
shell = new ShellComponent();
}
	6

shell.setCommand(bButton, false);	8
shell.addComponent(get_fForm());	6
return shell; }	4
private FormComponent get_fForm(){	4

if(fForm == null){
fForm = new FormComponent();
}
fForm.addComponent(get_lLabel0());
fForm.addComponent(get_tfComponent0());
	6

return fForm; }	4
private LabelComponent get_lLabel0(){	4

if(lLabel0 == null){
lLabel0 = new LabelComponent("이름");
}
	5

return lLabel0; }	4
private TextFieldComponent get_tfComponent0(){	4

if(tfComponent0 == null){
tfComponent0 = new TextFieldComponent("",
TextComponent.CONSTRAINT_ANY);
}
	5

return tfComponent0; }	4

public void action(Component cmp, Object o){
if(cmp == bButton){
}
}
	8

}	1

Ⅳ. 시스템 설계 및 구현

1. 시스템 설계

위피 응용프로그램 생성 시스템의 처리과정은 [그림 4]와 같다. 모바일 UI 디자인한 후, JIML을 자동으로 생성하고, 생성된 JIML을 위피 Jlet코드 변환 규칙을 이용하여 파싱과정을 거친 후, 자바 파일로 생성한다. 파싱 원리는 명세된 JIML을 분석하여, 요소 정보(structure, part, style, property, command, event)를 얻고, 이로부터 JIML에 명세된 UI객체와 이벤트 객체를 추출한 후, 이를 본 논문에서 제안한 위피 코드로의 변환 규칙을 이용하여 위피 소스 코드를 생성한다. 개발자는 생성된 코드를 기반으로 데이터 처리, 이벤트 처리 등의 추가 기능을 구현하여 위피 비즈니스 프로그램을 개발할 수 있다. 시스템은 위피 플랫폼을 대상으로 개발하고, C언어로 개발하는 Clet과 자바언어로 개발하는 Jlet플랫폼 중 자바의 장점인 재사용성의 증대를 위해 Jlet 플랫폼을 기반으로 구현한다.

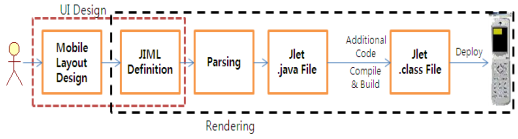


그림 4. 위피 응용프로그램 생성 시스템의 프로세서

위피 응용프로그램 생성 시스템 프로토타입의 클래스 구조는 [그림 5]와 같이 표현된다. JIML문서로부터 생성된 객체는 [그림 5]에 표현된 Part 클래스의 하위클래스와 JIMLHandler 클래스로 맵핑된다. 위피 응용프로그램 생성 시스템의 주요 클래스는 JIML 클래스, JIMLHandler 클래스, Part 클래스, Part 클래스를 상속받은 하위 클래스, 등으로 구성되는데, Part 클래스는 lwc 패키지내의 UI 컴포넌트 객체들을 대표하는 추상클래스로 class, id의 속성과 generateDeclare() 메소드, generateCreateMethod() 메소드를 선언한 추상클래스이다. 이로부터 상속 받은 ButtonComponent, TextFieldComponent, LabelComponent 등의 클래스는 JIML문서에 표현된 각 UI 요소에 해당하는 property, value 값을 가지며, Part 추상클래스의 추상메소드를 구현한다. 또한 Part 추상클래스의 class, id 속성으로부터 해당 컴포넌트 명과 id 값을 상속받는다. JIMLHandler 클래스는 DOM 파서로서 JIML의 태그 property, value 값을 추출하고, 가공 단계에서 해당 컴포넌트 객체를 생성하고, 추출된 값을 Vector에 저장한

다. JIML 클래스는 JIML문서를 추상화한 클래스로 모바일 화면으로부터 넘어온 UI 명세 정보를 토대로 JIML태그를 실시간으로 생성한다.

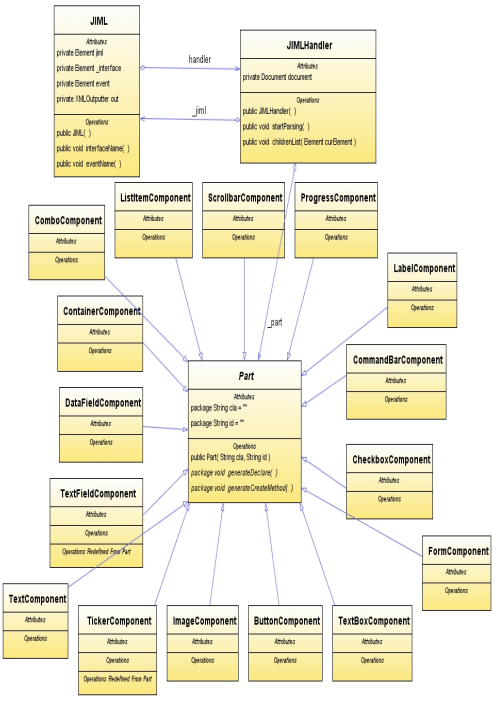


그림 5. 위피 응용프로그램 생성 시스템의 클래스 구조

소스코드 생성을 위한 과정은 3단계로 나눌 수 있는데, 먼저 위피 API의 상위레벨 컴포넌트 컨트롤로 모바일 화면을 디자인 하면, UI 명세 정보를 토대로 JIML태그를 생성하고, 이는 JIML 문서로 완성된다. 두 번째로 JIMLHandler 클래스에서JIML 문서를 파싱하여 JIML 객체로 변환하고, 세 번째로 변환된 JIML객체로부터 Jlet 소스코드를 생성한다. 두 번째 단계에서는 JIML 문서의 파싱을 통해 해당 UI 컴포넌트 클래스와 필드의 값을 결정하여 그 값을 Part 클래스의 필드 값에 저장한다. 세 번째 단계에서는 JIML 객체의 Part 클래스의 필드 값으로부터 제안한 변환 규칙을 적용하여 위피 코드를 생성한다. 이때 UI 객체의 generateDeclare() 메소드와 generateCreateMethod() 메소드를 수행하여 객체 정보를 가공하여 코드를 생성하게 된다.

2. 시스템 구현

시스템은 JDK 1.6과 Netbeans 6.1을 사용하여 개발하였으며, Jlet 실행환경은 AromaWIPI Emulator 1.1.18을 사용하였다. 위피 응용프로그램 생성 시스템의 실행화면은 [그림 6]과 같다. 먼저 [그림 6]의 왼쪽 그림과 같이 마법사 기능을 이용하여 문서 경로 및 이름, 클래스 명과 위피 classes.jar 파일경로를 입력하면, 오른쪽 그림과 같이 하나의 프로젝트가 생성된다.

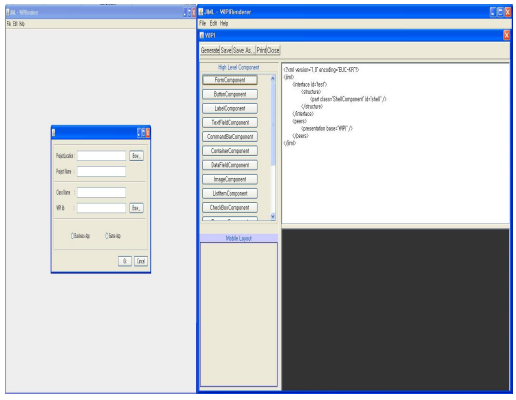


그림 6. 시스템의 마법사 기능 및 실행화면

시스템의 기능은 기존의 위피 프로그래밍 도구에는 없는 위피 API 상위레벨 컴포넌트를 비쥬얼화, 위피 플랫폼 UI화면 디자인 기능, 디자인을 통한 자동적인 JIML 문서 생성, JIML 문서를 소스코드로 변환하는 기능이 있다. 시스템의 실행 창은 네 개의 영역으로 나뉘어져 있는데, 왼쪽 상단은 위피 API의 상위레벨 컴포넌트를 컨트롤로 제공한 것이고, 왼쪽 하단은 위의 컨트롤을 활용하여 UI를 그릴 수 있는 모바일 레이아웃을 나타낸다. 오른쪽 상단은 JIML 문서, 오른쪽 하단은 위피 코드를 생성하는 텍스트 창을 나타낸다.

모바일 화면을 컨트롤로 디자인하기 위해서는 해당 컨트롤을 마우스로 선택한 후, 모바일 화면을 클릭을 하면[그림 7]과 같은 해당 컴포넌트에 관련된 마법사가 실행되어 쉽게 property, value 값을 설정할 수 있다. [그림 7]의 마법사 기능은 [표 3]의 Jlet Vocabulary에 나와 있는 property들을 나타낸다. 간단한 예로 TextFieldComponent를 모바일 화면에 디자인하고자

한다면, [그림 7]의 오른쪽 마법사가 실행이 되는데, TextFieldComponent는 Jlet Vocabulary에서 text, constraints라는 속성과 CONSTRAINT_NUMBER, CONSTRAINT_PASSWORD, CONSTRAINT_ANY 값을 가지고 있다. 이러한 property, value를 마법사를 통해 쉽게 입력할 수 있도록 마법사 기능이 구성되었다.

그림 7. 위피 API의 상위레벨 컴포넌트 마법사 기능

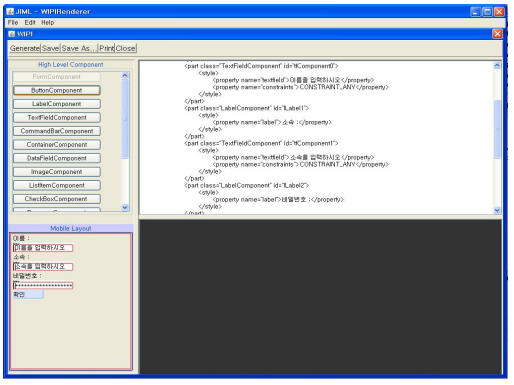


그림 8. JIML 문서 실시간 생성

마법사 기능을 이용하여 모바일 화면을 디자인할 때마다 [그림 8]의 오른쪽 상단의 JIML 텍스트 창에 실시간으로 태그 정보가 기록된다. 하지만, 위피 코드는 실시간으로 생성하지 않고, 최종적으로 모바일 화면 디자인이 완성된 후, 시스템의 상단 Generate 버튼을 누르면, JIML 태그 정보를 토대로 [그림 8]의 오른쪽 하단의 위피 코드 텍스트 창에 위피 코드가 생성된다. [그림 9]는 JIML 파싱을 통해 위피 코드를 생성하는 위피 응용프로그램 생성 시스템 최종 결과물을 보여주고 있다.

위피 언어의 경우 실제 직접 코딩에 의해서도 사용자 컴포넌트의 생성은 그 코드 길이가 길지 않지만, 다수의 위피 비즈니스 UI 객체를 개발하려면, 상당한 시간이 소요된다. 하지만, 본 논문의 자동화시스템을 이용하면 다수의 위피 UI객체를 빠르게 개발할 수 있다.

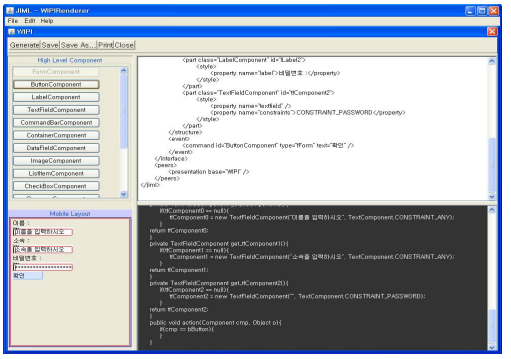


그림 9. 위피 코드를 생성하는 시스템의 최종 결과물

3. 사례연구

본 논문 결과물을 통해 생성되는 위피 응용프로그램 코드의 생산성을 평가하기 위해 비즈니스 프로그램의 일종인 CSP-WIPI[9] 응용프로그램의 UI를 개발하였다. CSP-WIPI는 모바일 쇼핑을 할 때 각각의 Client의 선호하는 사항을 미리 저장하여 그에 해당하는 상품만을 불러와서 최종 Client에게 보여주는 응용프로그램이다. 사례에서는 CSP-WIPI 응용프로그램의 UI를 논문에서 개발한 시스템으로 디자인하여, JIML 및 위피코드로 생성한다.

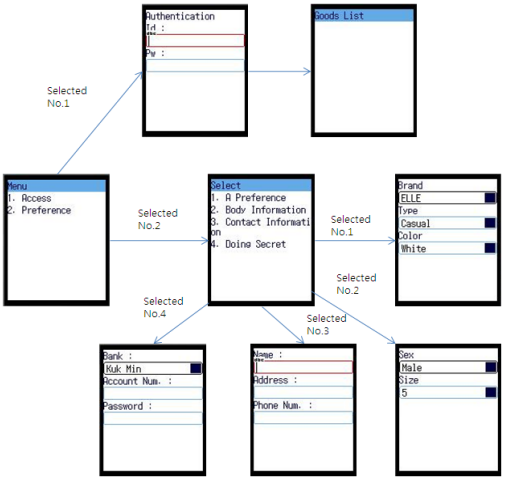


그림 10. 시스템을 이용하여 생성한 화면

CSP-WIPI 응용프로그램의 총 8개의 UI가 [그림 10]과 같이 정상적으로 생성되었다. CSP-WIPI 응용프로

그램의 총 UI관련 객체 수는 49개이고, 위피 코드로의 변환했을 때 생성된 위피 코드 라인 수는 총 545라인이 생성되었다. 실제로 CSP-WIPI 응용프로그램에는 쓰레드 처리, 키 이벤트 명령 등의 부가적인 코드가 추가되지만, 본 사례연구에서는 UI 생성에 관한 프로토타입 코드만을 생성한다. 사례연구 결과인 각각의 UI 객체 수, 코드 라인 수의 확인을 통하여 모든 UI 객체가 완벽하게 생성이 되고, 상당한 수의 UI 관련 코드를 빠르게 생성함을 확인할 수 있었다. 각 화면별 생성된 객체 수 코드 라인 수는 [표 6]과 같다.

표 6. 각 UI별 생성된 위피 코드 라인 수

UI 명	UI 객체 수	생성된 코드 라인 수
main	4	47
authentication	7	69
goods list	2	33
select	6	62
preference	8	96
body information	6	76
contact information	8	78
doing secret	8	84

V. 결론 및 향후 연구 방향

현재 위피 비즈니스 프로그램을 개발하기 위해서는 개발자가 텍스트 편집기를 이용하여 전체과정을 직접 손으로 타이핑하여 개발이 이루어지고 있다. 또한 기존의 편집기들은 위피 프로그래밍을 위한 비쥬얼 요소가 부족하다 이로 인해 개발시간이 길어지고, 모바일 문법에 미숙한 사용자는 위피 응용프로그램을 개발하기가 쉽지 않다. 본 논문에서는 위피 프로그래밍 자동화를 위해서 4단계의 개발 방안을 제시했으며, 이에 특화된 자동화시스템을 구현하였다. 또한 사례연구를 통하여 위의 개발 방법의 효율성을 증명하였다.

본 논문은 위피 API의 상위레벨 컴포넌트를 분석하여 위피 Jlet Vocabulary를 정의하였고, 이는 모바일 UI를 JIML로 명세할 때 활용된다. 또한 JIML의 요소와 스키마 구조를 정의하였다. 명세한 JIML을 소스코드로 변환하는 것은 신속한 위피 비즈니스 프로그램 개발을 위해서 중요한 요소이다. 이에 JIML 분석을 통해 위피 코드로 변환하기 위한 8가지 변환 규칙 또한 정의하였다. CSP-WIPI의 사례연구 결과 제안한 JIML은 위피 API의 상위레벨 UI 명세를 완벽히 지원하였다. 직관적인 XML 문서와 시각적 모델을 사용하여 위피 UI를 빠르게 생성할 수 있음을 위의 사례를 통해 알 수 있었다.

참 고 문 헌

[1] 유용덕, 박충범, 최훈, 김우식, “위피 응용프로그램 개발환경 설계 및 구현”, 한국정보처리학회 논문지 C, pp.749-756, 2005.

[2] 이동수, 박기창, 박승범, 김병기, “위피 콘텐츠 개발을 지원하기 위한 통합개발환경”, 한국정보처리학회 2007년 춘계학술발표대회, pp.160-163, 2007.

[3] 이동수, 김병기, “위피 콘텐츠 전용 저작도구 설계 및 구현”, 한국소프트웨어공학기술 합동 워크샵 2007, pp.72-76, 2007.

[4] 이동수, 박기창, 김철현, 이상준, 김병기, “GUI 위젯을 이용한 위피 코드 생성 저작도구 설계”, 한국정보처리학회 2008년 춘계학술발표대회, pp.331-334, 2008.

[5] 박기창, 서성채, 김병기, “J2ME MIDlet 사용자 인터페이스 자동생성을 위한 XML언어”, 한국정보처리학회 논문지 D, pp.327-336, 2008.

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

[7] http://www.alphaworks.ibm.com/tech/auiml

[8] http://www.ximl.org

[9] 김철민, 서성채, 유진호, 김병기, “Object Pool 패턴을 이용한 WIPI기반 MVC 모델의 개선”, 한국정보처리학회 2004년 추계학술대회, pp.307-310, 2007.

[10] 박상훈, 권혁주, 김영근, 이양선, “모바일 콘텐츠

의 재사용을 위한 GVM C-to-WIPI Java 변환기의 설계 및 구현," 한국정보처리학회 2006년 추계학술대회, pp.717-720, 2006.

저 자 소 개

이 동 수(Dong-Su Lee) 준회원

- 2005년 2월 : 호남대학교 전자과(공학사)
- 2007년 3월 ~ 현재 : 전남대학교 전자컴퓨터공학과(석사과정)

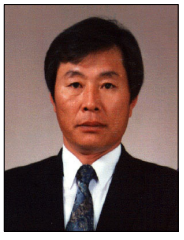
<관심분야> : CASE 도구, Mobile, 소프트웨어공학

박 기 창(Ki-Chang Park) 정회원

- 2001년 2월 : 전남대학교 물리학과(이학사)
- 2003년 2월 : 전남대학교 전산학과(이학석사)
- 2005년 2월 : 전남대학교 전산학과 박사수료

<관심분야> : CASE 도구, 소프트웨어 가시화

김 병 기(Byung-Ki Kim) 정회원



- 1978년 2월 : 전남대학교 수학교육과(이학사)
- 1980년 2월 : 전남대학교 수학과(이학석사)
- 2000년 : 전북대학교 수학과(이학박사)
- 1981년 ~ 현재 : 전남대학교 전자컴퓨터공학부 교수

<관심분야> : 소프트웨어공학, 객체지향시스템