무선인터넷플랫폼을 위한
사용자 인터페이스 컴포넌트 API 설계 및 구현

*연대진, 김연수, 박충범, 임형택, 최 훈
충남대학교 컴퓨터공학과
e-mail : {djyeon,kimys,cbpark,htlim,hchoi}@ce.cnu.ac.kr

Design and Implementation of the User Interface API On Wireless Internet Platform

*DaeJin Youn, YounSoo Kim, ChoongBum Park, HyongTaek Lim, Hoon Choi
Dept. of Computer Engineering, Chungnam National University

요 약

최근 무선인터넷 시장은 기능뿐만 아니라 사용자의 편의성과 디자인을 중시하고 있다. 따라서 이동통신사업자는 사용자 기호에 맞추기 위해 다양한 사용자 인터페이스를 제공하기 위하여, 구현 및 설계에 시간과 비용을 중복투자하고 있다. WIPI 에서는 이동통신사업자가 사용자 인터페이스를 효율적으로 구현할 수 있도록 사용자 인터페이스 컴포넌트(UIC)를 규정하고 있다. 본 논문에서는 UIC 의 확장성과 코드 재사용성을 위한 객체지향적 설계 및 구현 방법을 기술하고, 플랫폼 인증 툴킷과 테스트용 응용프로그램을 통한 사용자 인터페이스 컴포넌트를 검증하고, 적합성을 기술한다.

1. 서론

국내 이동통신사업자들은 각자 사업 여건에 알맞은 무선인터넷 플랫폼을 채택하여 소비자들에게 서비스를 제공해 왔다. 이로 인해 소비자가 원하는 컨텐츠를 사업자가 제공하지 않으면 사용할 수 없게 되는 문제점을 낳게 되었다. 따라서 한국무선인터넷표준화포럼(KWISF), 한국정보통신기술협회(TTA), 전자통신연구원(ETRI)은 국내 이동통신 3 사를 중심으로 플랫폼을 하나의 규격으로 통일하는 무선인터넷 표준 플랫폼 WIPI(Wireless Internet Platform for Interoperability) 1.0 을 2002 년에 제정하였고, 2004 년 9 월 현재 WIPI 2.0.1 을 제정하였다([1][2]).

무선인터넷 시장에서 상업적으로 성공하기 위해 기능 못지 않게 사용자 인터페이스가 중요하다. 따라서 이동통신사업자가 효율적으로 개발할 수 있도록 WIPI 에서는 사용자 인터페이스 컴포넌트(UIC : User Interface Component)를 제공하고 있다([3][4][5]).

UIC 는 텍스트 박스, 날짜/시간 컴포넌트, 메뉴 컴포넌트, 라벨 컴포넌트 및 리스트 컴포넌트로 구성된다. 각 컴포넌트는 응용프로그램에 포함되어 독립적으로 동작하며, 응용프로그램 개발을 용이하게 하는 역할을 수행한다. 또한 각 컴포넌트는 생성, 소멸, 페인팅 및 이벤트 핸들링이 필요하며, 이것은 WIPI C 응용프로그램과 같은 구조를 갖는다.

본 연구는 향후 확장성과 코드 재사용성을 위해 객체지향 개념을 도입하여 UIC 를 구현하였다. 따라서 C 언어로 구현하였지만 구조적 설계방법이 아닌 객체지향적 설계방법을 통하여 UIC 를 생성하였다([6]). 그리고 클래스에 모든 멤버함수 및 멤버변수를 포함한 형태가 아니라 주요 함수만을 클래스화 하고, 나머지는 API 를 통하여 제어하는 방식을 택했다. 이를 통해 확장성 및 효율성 등의 장점을 얻을 수 있었다.

본 논문은 2 장에서 UIC 를 소개하고, 3 장에서는 UIC 설계 및 구현 방법, 4 장에서는 플랫폼 인증 툴킷과 테스트용 Clet 을 통해 UIC API 를 검증하고, 적합성을 기술한다. 마지막으로 5 장에서는 결론 및 향후 연구방안을 기술한다.

2. WIPI UIC 용 C API

2-1. UIC 소개

WIPI에서의 UIC API는 무선인터넷 단말기의

* 본 연구는 한국전자통신원 "윈도우용 Clet 에뮬레이터 개발과제" 의 지원을 받아 수행되었음

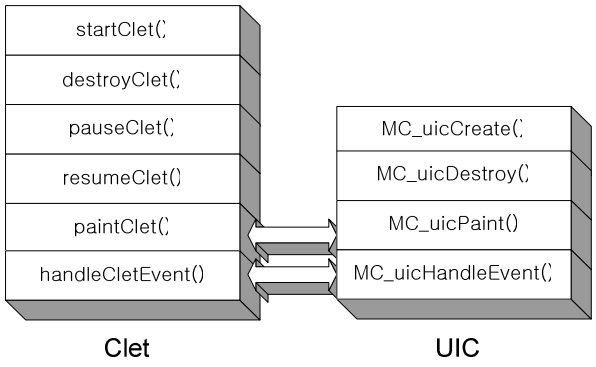
GUI(Graphic User Interface) 지원을 목적으로 한다. UIC는 응용프로그램에 포함되어 독립적으로 작동하며, 응용프로그램 개발을 용이하게 하는 역할을 수행한다. 모든 UIC는 루트 컴포넌트를 상속 받아서 생성되며, 텍스트 박스, 날짜/시간 컴포넌트, 메뉴 컴포넌트, 라벨 컴포넌트 그리고 리스트 컴포넌트로 구성된다. **<표 1>**은 WIPI C API 규격에 정의하고 있는 각 UIC 컴포넌트에 대한 설명이다.

<표 1> UIC API

구 분	설 명
루트 컴포넌트	최상위 컴포넌트
텍스트 박스 컴포넌트	루트 컴포넌트를 상속 받으며, 사용자에게 문자열을 보여주고, 해당 문자열을 변경할 수 있도록 해주는 컴포넌트
날짜/시간 컴포넌트	루트 컴포넌트를 상속받으며, 사용자에게 날짜/시간을 보여주고, 입력 받는 컴포넌트
메뉴 컴포넌트	루트 컴포넌트를 상속받으며, 화면에 메뉴를 출력하고, 사용자로부터 입력을 받음. 상하/좌우 키로 현재 선택 메뉴 항목을 변경하여 지정하도록 함.
라벨 컴포넌트	루트 컴포넌트를 상속받으며, 화면에 문자열을 출력해 줌. 크기에 맞도록 문자열의 줄을 바꿀 수 있음.
리스트 컴포넌트	루트 컴포넌트를 상속받으며, 화면에 리스트를 출력하고, 사용자로부터 입력을 받음. 상하/좌우 키로 리스트 항목을 변경하여 지정하도록 함.

2-2. UIC 내부 모듈

UIC는 기본적으로 생성, 소멸, 이벤트 처리 및 페인팅 모듈로 구성된다. **[그림 1]**과 같이 UIC 구조는 WIPI C 응용프로그램(Clet)의 구조와 같다([7]). 즉 UIC는 WIPI C 응용프로그램에 포함되어 독립적으로 작동하는 작은 응용프로그램과 같은 역할을 한다.



[그림 1] Clet 과 UIC 의 관계

- **MC_uicCreate()** : 컴포넌트 클래스 구조체를 매개 변수로 컴포넌트를 생성한다.
- **MC_uicDestroy()** : 생성된 컴포넌트를 파괴한다
- **MC_uicHandleEvent()** : 컴포넌트의 이벤트 처리를 담당한다. Clet의 handleCletEvent 함수 내에서 호출하여 처리된다. 또한 콜백 함수를 등록하여 특정 이벤트에 대한 기능을 제공할 수 있다.
- **MC_uicPaint()** : 컴포넌트를 디스플레이한다. Clet의 paintClet 함수 내에서 처리된다.

3. UIC C API 설계 및 구현

3-1. UIC API 의 객체지향 설계 및 구현

본 연구는 확장성 및 코드 재활용성을 위해 객체 지향 개념을 도입해 UIC를 설계 및 구현하였다.

가) 객체지향적 UIC의 캡슐화 설계

객체는 자신의 동작원리를 클래스로 캡슐화하며, 해당 객체만이 자신의 동작이 어떻게 작동하는지를 알고 있고, 외부에서는 알 수 없다. UIC에서는 클래스 구조체에 멤버 함수의 포인터를 저장함으로써 캡슐화를 구현했다. API 호출 시에 저장된 함수가 호출됨으로써, 외부에서는 같은 함수를 통해 원하는 동작을 얻을 수 있다. 그리고 부모 컴포넌트를 자식 컴포넌트 구조체의 첫 번째 항목으로 두어 캐스팅을 통해 멤버 변수의 액세스 범위 제한을 할 수 있다.

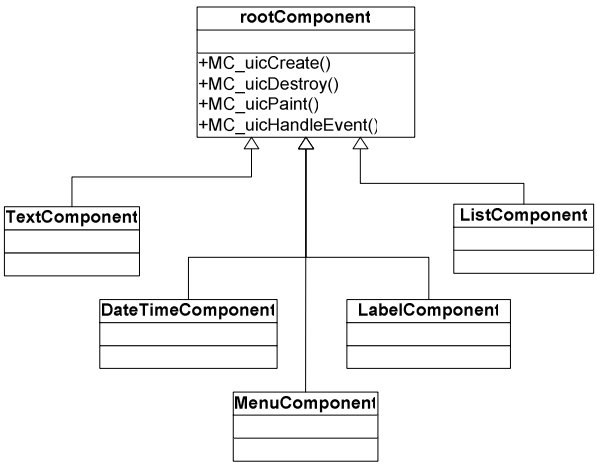
나) 객체지향적 UIC의 상속 설계

하나의 객체는 클래스의 인스턴스로서 부모 클래스 및 해당 클래스의 특성을 이어 받는다. 구현한 UIC에서는 클래스 구조체에 부모 클래스의 포인터를, 컴포넌트 구조체에 클래스 구조체의 포인터를 저장함으로써 상속성을 제공하였다. 그리고 부모 객체를 포함하여 객체를 생성함으로써 부모의 모든 특성을 포함하도록 하였다.

다) 객체지향적 UIC의 다형성 설계

같은 이름의 동작이 행위의 대상에 따라 다른 동작을 하도록 UIC에서는 클래스 안에 멤버 함수의 포인터를 저장한다. 그래서 생성, 소멸, 페인트 및 이벤트 핸들링을 수행할 경우 해당 클래스의 멤버 함수를 호출하도록 했다.

설계한 UIC 클래스 구조는 **[그림 2]**의 클래스 다이어그램과 같다.



[그림 2] UIC 클래스 다이어그램

UIC의 설계 및 구현에 있어서 컴포넌트 생성 시 중복

되는 부분을 부모 클래스화해서 코드의 재사용성을 늘리고, 다형성이 필요한 부분만을 클래스의 멤버함수화하여 코드를 최적화하였다. 자식 컴포넌트 고유의 제어는 멤버함수가 아닌 정의된 API를 통하여 제어함으로써 확장성, 효율성 등의 장점을 얻을 수 있었다.

본 연구에서 구현한 UIC의 각 컴포넌트의 객체지향적 설계 및 구현은 날짜/시간 컴포넌트(**[그림 3]**)의 구현 방법 같다.

```
struct _MC_UicClass clsTime = {
  "DateTimeComponent",
  &clsRoot,
  sizeof(MC_UicDateTimeComponent),
  dateCreate,
  dateDestroy
  datePaint,
  dateHandleEvent
};

struct _MC_UicComponent{
  MC_UicClass cls;
  MC_GrpFrameBuffer screen
  M_Int32 x
  M_Int32 y
  M_Int32 w;
  M_Int32 h;
  M_Int32 mask;
  M_Int32 font;
  M_Int32 pxlfg;
  M_Int32 pxlbg;
  MC_UicEventHandlerProc handler
  M_Int32 pcbId;
  M_Int32 extdata;
};

struct _MC_UicDateTimeComponent{
  MC_UicComponent cmpId;
  M_Int32 mode;
  M_Int16 select;
  time_t diff_time;
};
```

[그림 3] 날짜/시간 컴포넌트의 생성

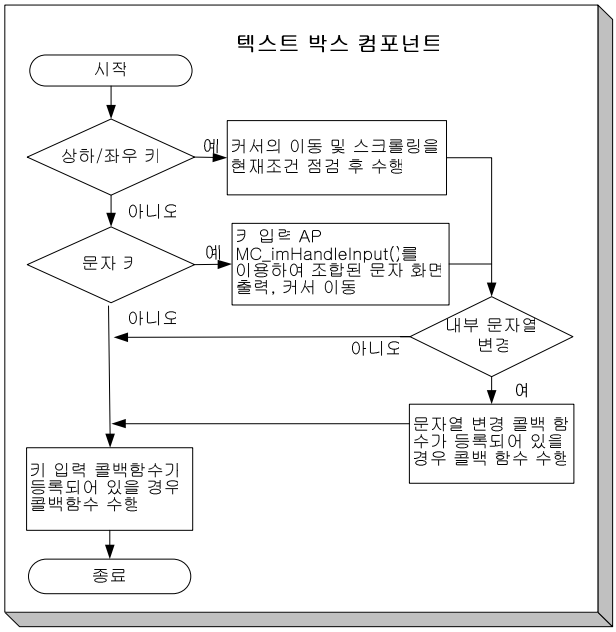
- **_MC_UicClass** : 전역변수로 미리 저장되어 있는 클래스의 정보를 담는 구조체이다. 클래스 이름, 부모 클래스 포인터, 사용 메모리 크기, 멤버함수 포인터를 저장한다.
- **_MC_UicComponent** : 루트 컴포넌트로서 _MC_UicClass 구조체의 포인터를 갖는다. 디스플레이를 위한 정보와 이벤트 핸들러, 콜백 함수의 포인터를 저장한다.
- **_MC_UicDateTimeComponent** : 날짜/시간 컴포넌트로서 루트 컴포넌트인 MC_UicComponent 구조체 포인터를 포함한다. 날짜/시간에 필요한 멤버 변수를 갖는다.

3-2. UIC API 세부 설계 및 구현

각 UIC는 disable, enable 상태에 따라서 이벤트 핸들링의 유무가 결정되며, 페인팅하는 방식이 다르다. 예를 들면 disable 상태에서 텍스트 박스는 커서를 표시하지 않으며, 메뉴와 리스트는 선택된 아이템을 강조하여 디스플레이하지 않는다.

- **텍스트 박스 컴포넌트**

[그림 4]는 텍스트 박스 컴포넌트의 키 입력에 대한 처리 과정을 나타낸다.



[그림 4] 텍스트 박스 컴포넌트 키 입력 처리

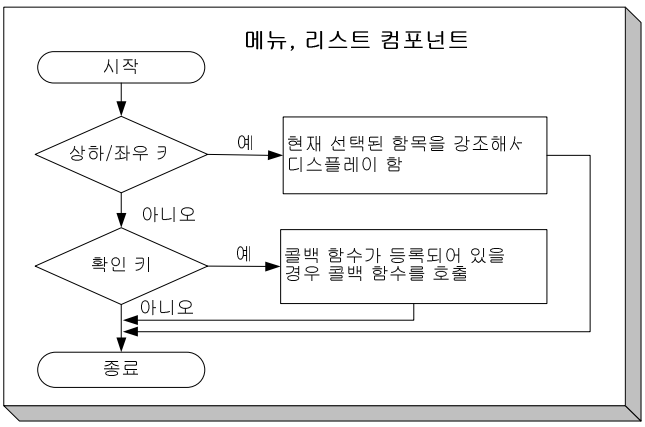
텍스트 스트링을 저장하는 버퍼는 일정 크기의 배열을 사용하여 구현하였다. 사용자의 입력이 버퍼의 배열 크기를 초과할 경우 힙(heap) 영역을 약속된 크기만큼 할당하여 텍스트를 저장하고, 또다시 초과 경우 다시 약속된 크기의 힙 영역 할당을 반복한다. 텍스트의 연결성을 보장하기 위해 힙 할당 시 포인터를 순차적으로 저장하였다.

- **라벨 컴포넌트**

이벤트 핸들러는 동작하지 않는다. 라벨의 스트링 길이를 통해 왼쪽 정렬, 가운데 정렬, 오른쪽 정렬 시 스트링의 시작 위치를 계산하였고, 줄 바꿈을 할 위치를 계산하였다.

- **메뉴, 리스트 컴포넌트**

[그림 5]는 메뉴, 리스트 컴포넌트의 키 입력에 대한 처리 과정을 나타낸다.



[그림 5] 메뉴, 리스트 컴포넌트 키 입력 처리

메뉴와 리스트 선택 시 순환하여 선택되며, 아이템을 어느 곳이든지 추가, 삭제할 수 있도록, 환형 이중 링크드 리스트(circular double linked list)를 이용하여 구현하였다.

● **날짜/시간 컴포넌트**

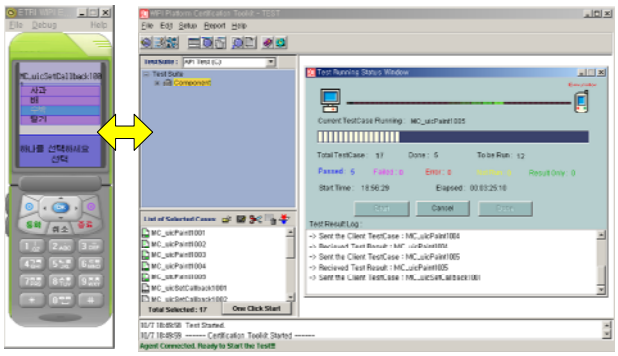
HAL(Hardware Abstraction Layer)에서 제공하는 OS(Operating System)의 시간과 유저가 설정한 시간의 차이를 저장하여 플랫폼 레벨의 시간을 만든다. 그리고 정형화된 포맷으로 사용자에게 날짜/시간을 디스플레이해준다.

4. WIPI 의 검증

4-1. PCT 를 이용한 UIC C API 검증

본 연구에서 구현한 UIC C API의 적합성은 EXEmobile ㈜에서 개발한 플랫폼 인증 툴킷인 PCT(Platform Certification Toolkit)를 통하여 검증하였다. PCT를 이용한 UIC C API의 검증 방법에는 batch test와 interactive test가 있다. batch test는 각각의 API에 미리 작성된 테스트 케이스를 통하여 API를 검증하는 것을 말하고, interactive test는 테스트 케이스의 수행결과를 테스트 요원이 확인하여 통과 또는 실패를 판단하는 것을 말한다.

[그림 6]은 에뮬레이터와 PCT 간의 통신을 통하여 테스트가 진행되는 과정을 나타낸다.



에뮬레이터 PCT

[그림 6] PCT 를 통한 검증

UIC API 테스트는 batch test 41 개 항목과 interactive test 16 개 항목으로 이루어지며, 본 검증 과정에서 모든 항목을 통과하였다.

4-2. 테스트용 Clet 을 통한 UIC C API 검증

실제 활용에서의 검증을 위해 본 연구는 WIPI 1.2 API 구현과 함께 개발한 AM(Application Manager)을 이용하였다. AM은 응용프로그램의 설치, 삭제 실행을 담당하는 최상위의 부모 프로세스의 역할을 하며, 구현에는 다양한 UIC가 사용되었다. 따라서 여러 응용프로그램을 실행하면서 스트레스 테스트를 통해, UIC API의 무결성을 검증하였다. **[그림 7]**은 AM을 통한 UIC API의 검증 모습이다.

[그림 7] AM 을 통한 UIC API 검증

5. 결론

본 연구에서 설계, 구현한 UIC 는 향후 확장성과 코드 재사용성을 위해 객체지향 개념을 도입하였다. 또한 클래스에 모든 멤버함수 및 멤버변수를 포함한 형태가 아니라 주요 함수만을 클래스화하고, 나머지는 API 를 통하여 제어하는 방식을 택했다. 이를 통해 확장성, 효율성 등의 장점을 얻을 수 있다. 구현한 UIC API 는 PCT 를 사용해 적합성을 검증하였고, 개발한 AM 을 이용하여 일반 응용프로그램에서도 문제가 없음을 입증하였다.

핸드폰의 기능이 추가됨에 사용자 인터페이스는 더욱 다양해지고 있다. 따라서 새로운 WIPI 규격이 제정됨에 따라 다양한 UIC 가 추가될 것으로 보인다. 따라서 객체지향적 UIC 는 새로운 컴포넌트 추가를 더욱 용이하게 할 것이다. 향후 연구방향은 WIPI 표준의 제정에 따라 새로운 버전의 UIC 개발을 계속 수행할 계획이다.

참고문헌

[1] 모바일 표준 플랫폼 WIPI 1.0, KWISFS.K-05-001

[2] 모바일 표준 플랫폼 WIPI 2.0.1, KWISFS.K-05-003

[3] 모바일 표준 플랫폼 WIPI 1.2.1, KWISFS.K-05-001R3

[4] “한국무선인터넷표준화포럼”, www.kwisforum.org, 2004.10.1

[5] “MOBILEJAVA Developer Community”, www.mobilejava.co.kr, 2004.10.1

[6] Axel-Tobias Schreiner, “Objektorientierte Programmierung mit ANSI C”, Hanser Fachbuch, 1994

[7] “윈도우용 Clet 에뮬레이터 개발”, ETRI 연구보고서, 충남대학교, 2004.8

[8] “휴대정보단말기용 웹 서비스 클라이언트 및 WIPI SDK”, ETRI 연구보고서, 충남대학교, 2003.11

[9] 유용덕, 김연수, 최훈, “동적 Upgrade 기능을 구비한 WIPI 개발환경 설계”, 한국통신학회 추계 종합 학술발표회 논문초록집, vol. 28, p.343, 2003.12