울산대학교 공학연구논문집 제30권 2호 pp. 125~138, 울산대학교.
Journal of Engineering Research Vol. 30, No. 2, pp. 125~138, 1999
University of Ulsan.

사용자 인터페이스 이전 방법에 관한 연구

이명재
컴퓨터·정보통신공학부

<요 약>

인터넷의 급속한 성장과 보급으로 인해, 웹 상에서 실행될 수 있는 소프트웨어에 대한 요구가 급증하고 있다. 이런 웹 응용 프로그램들을 전부 새롭게 개발하기보다는 기존의 응용 프로그램들을 새로운 환경에 맞게 재공학시켜주는 방법이 훨씬 효과적이다. 특히 사용자 인터페이스에 관련된 코드가 대화식 시스템의 반 이상을 차지한다는 사실에 비추어보면, 기존의 응용 프로그램들의 사용자 인터페이스를 자바 JFC에 맞게 효과적으로 재공학시키는 기술이 전체 재공학 프로세스를 크게 개선시킬 수 있을 것이다. 본 논문에서는 Motif 응용 프로그램들을 인터넷 환경으로 쉽게 이전할 수 있도록 기존의 Motif 사용자 인터페이스에서 자바 JFC 프레임웍으로 효과적으로 재공학하기 위한 방법을 제안한다. 제안한 방법은 사용자 인터페이스를 검출, 표현, 그리고 새로운 환경으로 변환하는 방법을 포함한다.

A Study on the Method for User Interface Migration

Myeong-Jae Yi
School of Computer Engineering and Information Technology

<Abstract>

Since Internet is very rapidly growing and spreading out currently, demands on the WWW applications are continuously increasing. However, it is more effective to reengineer the existing applications into the new environments than to develop from scratch these WWW applications. Considering the fact that half or more of the code in

an interactive system is devoted to implementing the user interface, the reengineering techniques for effectively migrating the graphic user interface of the existing systems into Java JFC could substantially improve the entire reengineering process. In this paper, we proposed a graphic user interface migration method for effectively translating Motif user interface into Java JFC framework to easily port Motif applications into WWW environments. Our method includes techniques for detecting and representing user interface components and transforming into new environments.

1. 서론

하드웨어의 급속한 성장과 함께 컴퓨팅 환경도 많은 변화를 하고 있다. 특히 네트웍 기술의 발전을 통한 인터넷의 급속한 성장과 보급은 소프트웨어 기술에 많은 변화를 요구하고 있다. 구조적 방법론, 객체 지향 방법론, 그리고 클라이언트 서버 기술 등으로 발전해온 소프트웨어 공학의 기술이 이제는 WWW과의 강력한 결합을 시도하고 있다. 사용자들은 특별한 클라이언트를 자신의 하드웨어에 설치하지 않고도, WWW 상에서 바로 소프트웨어를 실행하고 싶어한다.

이러한 WWW 브라우저 상에서 바로 실행 가능한 소프트웨어에 대한 해법으로 가장 주목받고 있는 것이 자바 프로그램이다. 자바 언어는 1995년에 SunWorld 학술회의에서 공식적으로 발표된 이후 놀라운 성장을 거듭하고 있으며, 많은 소프트웨어가 개발되어 WWW 상에서 운용되고 있다. 이러한 요구는 WWW의 성장과 함께 앞으로 더욱 증대될 것이 확실하다.

이런 WWW 운용 프로그램들을 전부 새롭게 개발하기보다는 기존에 개발되었던 많은 응용 프로그램들을 새로운 환경에 맞게 재공학시켜 주는 방법이 훨씬 경제적이고 효과적인 방법이다. 기존의 재공학 기술은 프로그램 이해, 자료 변환, 그리고 원시 코드 분석과 같은 방법을 사용하여 기존 프로그램의 기능을 추출하는 것에 초점을 맞추어 왔다[1].

그러나 많은 응용 프로그램들이 대화식으로 동작하며, 이런 프로그램에서 사용자 인터페이스에 관련된 코드는 전체 코드의 50% 정도를 차지하고 있다[2]. 이런 사실을 고려해 보면, 사용자 인터페이스에 대한 재공학 과정을 개선시키는 것에 의해 서로 다른 플랫폼간에 시스템을 이전하는 전체 작업을 크게 향상시킬 수 있다는 것을 알 수 있다. 그러나 아직까지는 사용자 인터페이스 재공학에 관련된 연구가 미흡한 현실이다.

사용자 인터페이스 재공학에 관한 기술은 단기적으로는 아직도 우리 주변에 산재해 있는 문자 중심의 사용자 인터페이스를 그래픽 사용자 인터페이스로 이전하는 방법에 대한 연구가 주류를 이루면서, PC와 Unix의 대표적 GUI 환경 사이의 호환에 관한 연구가 진행될 것으로 예측되며, 중장기적으로는 모든 사용자 인터페이스 사이의 호환성 문제를 해결하고자 하는 노력, 그리고 WWW 기반의 사용자 인터페이스로의 재공학 방법의 자동화에 대한 연구가 널리 진행될 것이다[3].

본 논문에서는 기존의 그래픽 사용자 인터페이스 기술을 사용하여 개발된 X 응용 프로그램들을 WWW 상에서 운용할 수 있도록 Java JFC 프레임웍으로 변환하는 재공학 방법을 제안한다.

2. 사용자 인터페이스 이전

2.1 용어

재공학에 관련된 용어들은 종종 여러 의미로 사용되며, 잘못 사용되어진다. 이 절에서는 사용자 인터페이스 이전에 관련된 여러 용어들의 의미를 정리한다. Chikofsky와 Cross는 재공학에 대한 전문 용어들에 대한 분류법을 제시하였고, 본 논문에서는 이 것을 기초로 한 다음과 같은 정의를 사용한다.[4].

- 이전(Migration): 이것은 하드웨어 플랫폼, 운용 환경, 또는 구현 언어를 포함하는 본래의 환경으로부터 새로운 환경으로 소프트웨어를 이동하는 활동이다.
- 재공학(Reengineering): 이것은 소프트웨어를 재구조화(restructuring), 재설계, 혹은 다시 구현하는 것을 포함한다. McClure는 재공학을 유지보수성(maintainability)를 향상시키고, 기술 수준을 높이며, 소프트웨어의 수명을 연장하고, 표준에 맞추기 위해 새로운 기술을 적용하여 기존의 시스템을 개선하는 것이라고 정의하였다.
- 이식(Porting): 이것은 프로그램을 언어의 구문과 운영 체제에 대한 인터페이스만을 변경하여 한 환경에서 다른 환경으로 이동하는 것을 의미한다. 이론적으로 이것에 대한 최소한의 방법은 프로그램을 다른 환경에서 다시 컴파일하는 것이다. 그러나 일반적으로 이식은 새로운 환경에 맞게 코드의 일부를 변경하는 것을 포함한다.
- 역공학(Reverse Engineering): 이것은 기존의 레거시(legacy) 시스템의 원래의 설계를 추상적인 표현으로 기술하기 위해 시스템을 분석하는 작업이다. 추상화된 표현물은 원시 코드와 존재하는 문서들에 대한 분석으로부터 유도된다. 역공학의 목적은 시스템에 대해 문서화를 다시 하여 그 프로그램에 대한 이해를 돕는 것이다.
- 순공학(Forward Engineering): 이것은 추상적 단계(분석 단계)와 설계 단계의 수준에서 구현 단계 수준으로 이동하는 것으로 일반적인 프로그램 개발 방향과 같다.

2.2 사용자 인터페이스 이전에서의 고려사항

어떤 환경의 사용자 인터페이스를 새로운 환경에 맞게 이전하는 문제는 단순하게 생각한다면 원래 환경의 사용자 인터페이스 구축 라이브러리들을 새로운 환경의 라이브러리들로 대응시키는 것에 의해 간단히 해결될 수 있으리라고 생각하기 쉽다. 그러나, 이 접근법은 몇 가지 문제점을 갖고 있다; 가장 커다란 문제점은 불완전성(incompleteness)에 관한 문제로서 원래 환경에 있던 기능이 새로운 환경에는 없는 경우, 혹은 원래 환경에 있던 것보다 더욱 많은 기능을 새로운 환경에서 제공하는 경우(예를 들면, OSF/Motif에서는 토글 버튼의 상태는 2가지이고, MS-Window에서는 3가지 상태를 가짐) 모두 임의적인 선택에 대한 문제가 발생한다. 다른 것으로는 상속성(inheritance) 계층의 차이점이나 콜백(callback)의 처리 방법의 차이와 같은 구조적인 문제점을 들 수 있다.

사용자 인터페이스를 이전하는 것이 어려운 또 하나의 이유는 사용자 인터페이스의 통합에 따른 문제로서, 많은 시스템들에서 사용자 인터페이스는 시스템의 다른 기능들과 통합되어 있기 때문에 이들을 인식하고, 시스템의 다른 부분들과 분리하여 내는 것이 쉽지 않다는 것이다.

지금까지 발표된 그래픽 사용자 인터페이스 생성기(GUI Generator), 추상 프로그래밍 인터페이스(API, Abstract Programming Interface), 라이브러리 치환 기법 등과 같은 기존의 사용자 인터페이스 재공학에 관련된 기술들은 서로 다른 플랫폼에 대한 지원이 부족하고 사용자 인터페이스 부품들이 도구들에 의해 임의로 지정되며, 생성된 인터페이스에 대한 조절이 매우 힘든 문제점들을 갖고 있다[4].

결국 하나의 환경의 사용자 인터페이스를 다른 환경의 사용자 인터페이스로 이전하는 것의 핵심 기술은 원래 시스템 내에서 어떻게 사용자 인터페이스 부분들을 인식하는 방법과 그렇게 인식된 부분을 불완전성의 문제와 구조적인 문제점들을 해결해 가면서 새로운 환경에 맞게 변환해 나가는 방법이다.

2.3 문자 중심에서 그래픽 사용자 인터페이스로의 이동 방법

독일의 Michael Schulz 등은 메인프레임 어플리케이션의 문자 기반 패널을 그래픽 사용자 인터페이스로 변환하기 위한 통합 도구 환경을 구축하고 있다. 이들은 패널 영역에 대한 일반화된 기술문을 사용하여 패널내의 다이알로그 개체의 인식에 근거한 문자 기반 패널의 정적 분석에 대한 접근 방법을 제시하였고, 인식 과정을 지원하는 도구를 개발하였다. 이들이 개발한 도구는 기존의 이전 도구에 비해 이전 과정의 자동화 정도를 향상시켰다[5].

Canada의 McGill 대학 CRIM 연구소에서는 Ettore Merlo 등이 사용자 인터페이스를 재

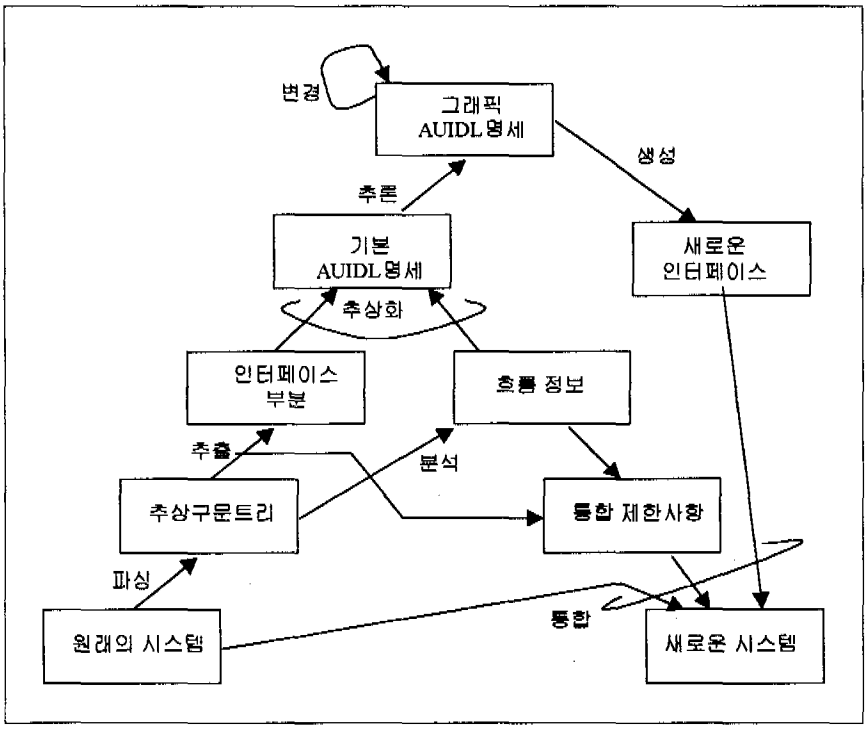

그림 1 Merlo의 사용자 인터페이스 재공학 모델

공학하는 모델을 제시하였다(그림 1). 그림 1에 나타낸 것처럼 이들은 구문 분석을 통해 사용자 인터페이스에 해당되는 부분을 인식하고, 이를 사용자 인터페이스의 구조와 행위를 기술할 수 있는 중간 표현 언어인 AUIDL(Abstract User Interface Design Language)로 표현한 후 추론 과정을 거쳐서 새로운 인터페이스를 생성하는 모델을 제시하였다. 이들은 이 방법을 사용하여 문자 중심의 COBOL/CISC 사용자 인터페이스 부분을 인식하고 이를 PC 기반의 새로운 그래픽 사용자 인터페이스 환경으로 재공학시키는 것에 이용하였다. 이들은 원시 코드로부터 추출된 사용자 인터페이스에 대한 내용은 프로세스 대수 (CCS)에 근거한 추상 행동 양식으로 변환하였다. 분석 작업의 일부분은 Refine/COBOL을 사용하여 수행되었고 변환 작업 자체는 수작업으로 수행하였다[6][7][8].

Italy의 IRST 연구소는 Canada의 Ecole Polytechnique와의 공동 연구를 통해 BASIC 언어로 작성된 문자 중심 사용자 인터페이스를 Visual C++ 의 그래픽 사용자 인터페이스로 바꾸는 방법을 발표하였다[9]. 이들의 재공학 프로세스는 다음과 같다. 먼저 Turbo Basic 원시 코드를 파싱하여 추상 구문 트리(Abstract Syntax Tree, AST)를 생성한다. AST는 상호작용에 대한 특정한 패턴 즉 상호작용 cliche를 인식하는 것에 사용되며, 또한 제어 흐름 그래프를 생성하는 데에도 이용된다. 인식된 cliche에 대응하는 추상 사용자 인터페이스 객체를 형성한다. 추상 사용자 인터페이스 클래스는 메뉴 선택, 버튼, 대화상자 등과 같이 그래픽 사용자 인터페이스에서 공통적으로 사용되는 상호작용 기법에 대한 개념적인 기술을 제공한다. 추상 사용자 인터페이스 객체와 제어 흐름 그래프를 사용하여 Visual C++ 의 그래픽 사용자 인터페이스 객체와 대응하는 콜백 코드를 생성한다. 이들의 방법은 캐나다 McGill 대학에서 사용한 방법과 유사하다.

미국 Texas Austin 연구소의 Larry Van Sickle 등은 COBOL 언어로 작성된 미니컴퓨터 응용 프로그램을 IBM 메인프레임상의 CICS에서 수행될 수 있도록 변환하는 방법을 개발하였다. 그들은 COBOL 원시코드를 Prolog 로 변환하는 방법을 사용하여 사용자 인터페이스 블록을 인식하고 휴어리스틱을 사용하여 사용자 인터페이스를 변환시켰다[10].

헝가리의 Laszlo Csaba는 대규모의 문자 기반의 오래된 DOS 응용 프로그램을 현대의 윈도우용 응용 프로그램으로 변환하는 프로젝트에서 사용된 사용자 인터페이스 재공학 방법을 발표하였다. 원래의 프로그램은 메인프레임 환경을 위해 COBOL로 작성되었다. 변환 작업중 핵심 운용 부분은 변경하지 않고 그대로 두었다. 이들이 적용한 방법에서 원래의 프로그램 - 변경된 입출력 호출을 갖고 윈도우용으로 재컴파일된 - 은 새로운 랩핑 (Wrapping) 응용 프로그램에 의해 통신 모듈을 통해 수행되고 원격 제어된다. 새로운 프로그램인 Artificial User 프로그램은 매우 효과적인 비주얼 도구인 S-Prog를 사용하여 개발하였고 원시 코드의 일부분은 Word Basic 마크로에 의해 생성하였다[11].

2.4 그래픽 사용자 인터페이스간의 이전

미국 Georgia 공대의 Melody Moore 등은 지식 표현 언어인 CLASSIC을 사용하여 PC 기반의 그래픽 사용자 인터페이스를 Workstation 기반의 그래픽 사용자 인터페이스 환경으로 바꾸는 연구를 수행하였다. 그들은 CLASSIC으로 MS-Windows와 Motif의 사용자 인터페이스 위젯(widget)에 대한 지식을 표현하였고 이를 그래픽 사용자 인터페이스 변환에 사용하였다. 이들은 레거시 시스템에서 사용자 인터페이스 부분을 찾아내기 위해 규칙

베이스를 개발하였다[12][13]. 이들은 또한 레거시 시스템에서 사용자 인터페이스 부분들을 찾아낼 수 있는 언어에 독립적인 규칙들의 집합을 정의하고 C, pascal, Ada, 그리고 Cobol 등과 같이 각각 다른 언어들로 작성된 응용 프로그램들의 문자 중심 사용자 인터페이스를 그래픽 사용자 인터페이스로 이전하는 작업을 수행하였다. 이들의 작업은 부분적으로 자동화되었으며, 함수 포인터 등의 문제점을 해결하기 위해 동적 분석의 필요성에 대해 언급하였다[14].

3. 사용자 인터페이스 이전 프로세스

사용자 인터페이스 이전을 하기 위한 프로세스는 크게 3가지 단계로 구분된다:

- 검출(detection) - 소스 코드에 대한 분석과 다른 기술들을 사용하여, 기존의 코드가 포함하고 있는 사용자 인터페이스 기능을 인식하는 것이다.
- 표현(representation) - 사용자 인터페이스를 검출한 후에, 그것의 기능을 묘사하고 문서화하는 것이다.
- 변환(transformation) - 새로운 환경에 대한 사용자 인터페이스를 생성하기 위해 매핑(mapping)을 수행하는 것이다.

본 장에서는 이들에 대해서 기술한다.

3.1 검출 단계 - 사용자 인터페이스 부분에 대한 인식

시스템을 이전할 때 발생하는 어려움의 대부분은 존재하는 설계를 이해하는 것이다. 사용자 인터페이스 이전에서 중요한 것은 사용자 인터페이스를 구현한 어플리케이션의 모듈 혹은 컴포넌트를 검출하는 것이다. 특히 사용자 인터페이스 기술이 완전한 재공학을 필요로 하거나 혹은 사용자 인터페이스의 변환을 필요로 할 때는 더욱 중요하다.

검출은 몇 가지 방법으로 수행되어 질 수 있다. 하나의 방법은 기존 코드의 호출 트리 혹은 데이터 흐름도를 생성한 후에 추이적(transitive)으로 그룹화하여 '사용자 인터페이스'로 분류될 수 있는 코드 세그먼트들을 인식하는 것이다. 다른 방법은 코드내에 있는 콜백의 위치를 찾아낸 후에 잠재적인 사용자 인터페이스 객체를 인식하기 위해 이것들을 사용하는 것이다. 검출에 사용될 수 있는 방법들은 다음과 같다.

3.1.1 패턴 매칭에 대한 사용자 인터페이스 인식

구문 분석기에 의해 생성된 추상 구문 트리(AST, Abstract Syntax Tree)에서부터 사용자 인터페이스 부분을 인식하는 방법으로서, 코드 내에서 사용자 인터페이스의 구문적 패턴과 일치하는 부분을 시작점으로 하여 제어 흐름 분석(control flow analysis)을 사용하여 사용자와의 상호 작용에 대한 모드와 수행 조건에 대한 것을 인식하는 방법이다.

3.1.2 구문적/의미적 분석에 의한 사용자 인터페이스 인식

입력과 출력에 대한 기준들에 따라서 코드를 분석하여 COBOL 코드로부터 사용자 인터페이스 부분들을 인식하는 방법으로, ACCEPT 문장을 인식하고 그 지점에서부터 사용자와의 작업에 대한 것을 그룹화한다. 이 방법은 코드의 구조가 좋지 않을 때는 올바른 결과를 얻지 못할 수가 있다.

3.1.3 정적/동적 분석에 의한 인식 방법

함수에 대한 포인터의 사용과 여러 부분으로 나누어진 스크린에서 한 영역에 대한 입력이 다른 영역에 대한 입력과는 다른 의미를 가질 수 있으므로 이를 인식하기 위해 동적인 분석도 수행하는 방법이다.

3.2 표현 단계 - 인식된 부분에 대한 표현 방법

이 단계에서는 검출 단계에서 인식된 사용자 인터페이스 부분들을 추상화된 표현 양식으로 기술하는 것이다. 이 작업은 어떤 특정한 디스플레이 기술에 종속되지 않는 방법으로 시스템의 기능을 표현할 수 있어야 한다. 또한 사용자 인터페이스의 기능적인 요구사항을 전부 적절하게 표현하기에 충분할 만큼 완전하고 견고하게 수행해야 한다. 이런 표현 문제를 해결하고 모델을 만드는 것이 재공학 프로세스를 이해하는 데 있어서 핵심이다. 추상적인 표현물을 고안하는 것은 자동화된 도구들과 같은 보다 나은 재공학 지원을 개발하기 위한 기초가 된다. 일반적인 수준의 추상물을 표현하기 위한 방법으로는 다음과 같은 것이 있다.

3.2.1 추상 기술 언어

추상 기술 언어(Abstract Description Language)는 사용자 인터페이스의 명세를 위한 중간적인 표현으로서, 객체 지향 기법을 기반으로 하여 사용자 인터페이스 구조를 기술하고 프로세스 대수(algebra)를 이용하여 사용자 인터페이스의 행동 양식에 대한 것을 명세화한다.

3.2.2 유한 상태 기계

대부분의 사용자 인터페이스들이 시스템의 상태와 사용자의 입력에 의해 발생되는 천이를 다룬다는 것을 근거로 하여 사용자 인터페이스를 유한 상태 기계(Finite State Machine)로 표현하는 방법이다. 사용자의 선택에 의해 시스템의 상태가 바뀌어지는 것과 같은 메뉴들 사이의 천이를 기술하는 것에 효과적인 방법이다.

3.2.3 프롤로그 추상 구문 트리

COBOL 코드를 추상 구문 트리의 역할을 수행하는 프롤로그로 변환하는 것에 의해 사용자 인터페이스 구조를 표현하는 방법이다. 그 다음에는 프롤로그를 재구조화하고, 제어

흐름과 자료 구조에 대한 정보, 사용자 인터페이스에 대한 고수준의 기술문을 제공하기 위해 프롤로그를 조정한다.

3.2.4 객체 지향 표현

사용자 인터페이스를 기술하기 위해 객체 지향 자료 모델을 사용하는 방법으로, 사용자 인터페이스 객체들과 속성들을 표현하기 위해 지식 기반의 표현법을 사용한다. 사용자 인터페이스에 대한 행위를 명세화하기 위해 전제 조건과 사후 조건을 정의한다[15].

3.3 변환 단계 - 새로운 환경으로의 변환 방법

사용자 인터페이스 이전 프로세스의 마지막 단계는 기존의 시스템에서 새로운 시스템으로의 변환에 대한 여러 집합들을 고안하는 것이다. 변환 단계는 단순히 하나의 사용자 인터페이스 객체를 다른 것으로 번역하는 것 이상이다; 변환은 또한 다른 사용자 인터페이스 환경에 분명하게 매핑되지 않는 사용자 인터페이스 부품들에 대해 최상의 대응물을 판단하기 위해 의사-결정(decision-making)과 추론을 필요로 한다. 변환 모델을 개발한 후에는 이것을 새로운 사용자 인터페이스 환경에 적용한다. 변환 방법으로는 다음과 같은 것이 있다.

3.3.1 지식 기반의 변환 방법

사용자 인터페이스 부분들에 대한 지식 기반의 표현 방법을 사용하며, 이를 이용하여 새로운 환경에 맞게 변환하는 방법으로, 부품들을 기술하기 위해 CLASSIC 지식 표현 시스템을 사용하며, 수집된 자료들에 대한 추론 작업을 이용하여 개발된 매핑 방법이다.

3.3.2 상태 기계를 이용한 매핑 방법

유한 상태 기계를 이용하여 시스템을 기술한 후에, 특정한 사용자 인터페이스 부분들에 대한 상태와 천이 사이와 사용자 인터페이스 환경의 행동 사이를 매핑시키는 것에 의해 변환 작업을 수행하는 방법으로서, 유한 상태 기계에서의 상태들은 사용자들에 의한 선택 혹은 메뉴들을 나타내며, 천이들은 사용자의 입력 즉 선택을 나타낸다.

4. OSF/Motif에서 Java JFC로의 사용자 인터페이스 이전 방법

OSF/Motif에서 Java JFC로의 성공적인 사용자 인터페이스 재공학을 위해 해결해야 할 기술적 문제는 앞서 언급했듯이 X 응용 프로그램의 대표적인 그래픽 사용자 인터페이스인 Motif 응용 프로그램의 사용자 인터페이스의 인식에 관한 문제와 Motif에 대한 지식과 목적 응용 프로그램이 가져야 하는 Java JFC에 관한 지식을 어떻게 이용해서 불완전성과 구조적인 문제점들을 해결해야 하는 가이다.

4.1 OSF/Motif 어플리케이션에서 사용자 인터페이스 부분 인식

Motif 그래픽 사용자 인터페이스의 규칙과 기능성에 대한 분석 작업을 수행한 결과 Motif 응용 프로그램에서 사용자 인터페이스에 관한 원시 코드 부분을 인식하고 추출하기 위한 방법은 다음과 같다.

Motif는 사용자 인터페이스 부품들이 위젯(혹은 가젯)들로 구성되어있고, 이들은 크게 복합 위젯과 프리미티브 위젯으로 구성되어 사용자 인터페이스를 구성하게 된다. 위젯 또는 가젯을 생성하는 작업에 사용되는 함수들을 표 1에 표시하였다.

표 1의 좌측은 Xt Intrinsics의 함수들 중에서 위젯을 생성할 때 사용되는 함수들이고, 우측은 OSF/Motif 라이브러리의 함수들 중에서 위젯을 생성할 때 사용되는 함수를 나타낸다.

Xt Intrinsics	OSF/MOTIF
XtCreateManagedWidget() XtCreatePopupShell() XtCreateWidget()	XmCreate-*whatever*
XtVaCreateManagedWidget() XtVaCreatePopupShell() XtVaCreateWidget()	XmVaCreateSimpleCheckBox() XmVaCreateSimpleMenuBar() XmVaCreateSimpleOptionMenu() XmVaCreateSimplePopupMenu() XmVaCreateSimplePulldownMenu() XmVaCreateSimpleRadioBox()

표 1 위젯 생성 함수

OSF/Motif 라이브러리의 위젯 생성 함수인 XmCreate-whatever는 XmCreateMenuBar() 또는 XmCreateLabelGadget과 같이 whatever에 자기가 생성하고 싶은 위젯 또는 가젯의 이름을 포함하는 함수들을 나타내는 것으로서 OSF/Motif Release 1.2에는 53개의 위젯 생성 함수와 6개의 가젯 생성 함수들이 존재한다[16][17].

또한 사용자의 입력에 따른 동작들은 콜백과 이벤트 핸들러, translation 관리자 등을 통하여 처리하고 있다. 표 2는 Xt Intrinsics Release 5와 Motif Release 1.2 의 함수들 중에 이벤트와 콜백에 관련된 함수들을 정리한 것이다. 표 2의 좌측 위 부분에 있는 함수들은 이벤트 처리기를 등록할 때에 사용되는 함수들로서, widget, event_mask, nonmaskable, proc, client_data의 매개변수들을 가지고 있다. 한편, 표 2의 우측 위 부분에 있는 함수들은 콜백 함수를 등록할 때에 사용되는 함수들로서, widget, 콜백 유형, 콜백 함수와 같은 매개변수들을 가지고 있다[16]. 그리고, 표의 하단에 있는 함수들은 이벤트 처리기 또는 콜백 함수를 삭제할 때에 사용되는 함수들로서, 이들이 갖고 있는 매개변수는 등록하는 함수들의 매개변수와 같다.

사용자 인터페이스 인식에서 중요한 것은 각각의 위젯들의 생성/소멸과 관련된 부분과 콜백 등과 관련된 부분들 즉 사용자 인터페이스 부분들을 인식하여야 하는데 이러한 코드 대부분이 함수 포인터나 변수를 통해 처리하고 있기 때문에 이 문제를 처리하기 위해서 함수 매개 변수에 대한 처리 방법을 사용한다.

이벤트	콜백
XtAddEventHandler() XtAddRawEventHandler() XtInsertEventHandler() XtInsertRawEventHandler()	XtAddCallback() XtAddCallbacks()
XtRemoveEventHandler() XtRemoveRawEventHandler()	XtRemoveCallback() XtRemoveCallbacks()

표 2 이벤트/콜백 관련 함수

4.2 Motif 사용자 인터페이스 부분에 대한 지식 표현 방법

추출된 사용자 인터페이스 부분에 대한 지식을 표현하고 저장하기 위한 방법은 3.2에서 설명하였다. 이러한 방법 중에서 유한 상태 기계를 이용하는 방법은 그래픽 사용자 인터페이스의 행위를 표현하는 데는 적합하지만, 크기가 커질 경우 제어하기가 어려운 문제점을 갖고 있다. 또한 프롤로그 추상 구문 트리를 이용하는 방법은 굳이 프롤로그를 사용할 필요가 없었다. 따라서 인식된 그래픽 사용자 인터페이스를 효과적으로 표현할 수 있는 중간 언어로서 유한 상태 기계를 사용하는 방법만큼 행위를 잘 기술할 수 있는 추상 기술 언어를 사용한다.

추상 기술 언어의 문법 중의 일부를 그림 2에 나타내었다. 이것으로 추출한 Motif GUI 부분에 대한 기술뿐만 아니라 Java JFC GUI에 대한 기술(description)을 통해 둘 사이의 개념적 계층도를 개발할 수 있을 것이다.

4.3 Java JFC로의 변환 방법

Moitf와 Java JFC의 GUI 구성요소들 사이의 계층 구조와 역할, 둘 간의 공통점과 구조적인 문제점, 그리고 불완전성에 대한 문제를 해결할 수 있는 지식을 이용한 변환 방법이 필요하다.

이를 위해서는 불완전성과 구조적인 차이로 인한 문제점을 처리할 수 있도록 대부분의 그래픽 사용자 인터페이스가 가지는 부품의 공통적인 특성을 파악하였다. 이러한 공통적인 특성으로는 대체로 프리미티브, 관리자, 메뉴, 그리고 대화상자의 4종류로 부품들을 구분할 수 있다.

프리미티브는 텍스트, 이미지, 리스트, 선택 버튼, 스크롤바 등과 같은 가장 기초적인 요소들을 포함하며, 관리자는 프레임, 스크롤드 윈도우, 로우칼럼 등과 같이 프리미티브와 다른 관리자들을 포함할 수 있는 부품들을 나타낸다. 그리고 메뉴는 메뉴 항목과 메뉴 판, 분리자, 팝업 메뉴 등을 포함하며, 마지막으로 다이알로그는 에러, 경고, 프롬프트, 정보, 질의, 메시지, 그리고 파일 선택 대화상자 등을 포함한다.

그러므로 Motif에서 Java JFC로의 그래픽 사용자 인터페이스 변환은 이러한 공통적인 특성에 기초하여 두 인터페이스들 간의 개념적 계층도를 구성하여야 한다. 이런 개념적 계

```
paragraph          → section-header section-option end
section-header     → class class-id
                     | subclass subclass-id ':' class-id
                     | instances instance-id ':' class-id
section-option     → [contained-by-section] [contain-section] [constant-section]
                     [import-section] [attribute-section] [export-section]
                     [parameter-section] [port-section] [behavior-section]
contained-by-section→ contained by instance-id {';' instance-id }
contain-section    → contains instance-id {';' instance-id }
import-section     → import importation-item {';' importation-item }
attribute-section  → attribute declaration-item {';' declaration-item }
export-section     → export exportation-item {';' exportation-item }
parameter-section→ parameter parameter-item {';'  parameter-item }
port-section       → port port-item {';' port-item}
behavior-section   → behavior behavior-item {';' behavior-item }
```

그림 2 사용자 인터페이스를 표현하는 추상 기술 언어의 문법 중의 일부

층도의 대략의 모양을 그림 3에 표시하였다. 또한 그래픽 사용자 인터페이스와 통합되어 있는 다른 기능적 요소들을 모듈화하는 방법으로는 재모듈화 방법을 사용한다.

본 논문에서 제안하는 Motif 어플리케이션의 그래픽 사용자 인터페이스를 Java JFC로 이전하기 위한 전체적인 모델을 그림 4에 나타내었다.

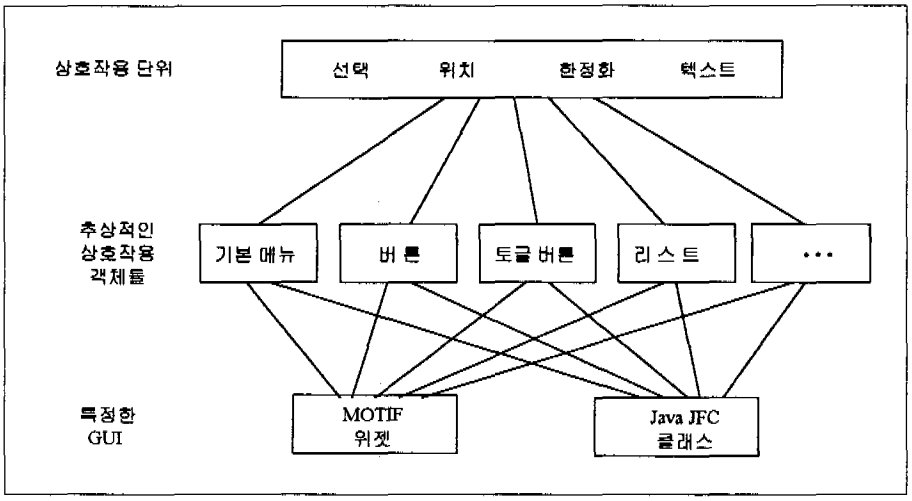

그림 3 사용자 인터페이스 변환을 위한 개념적 계층도

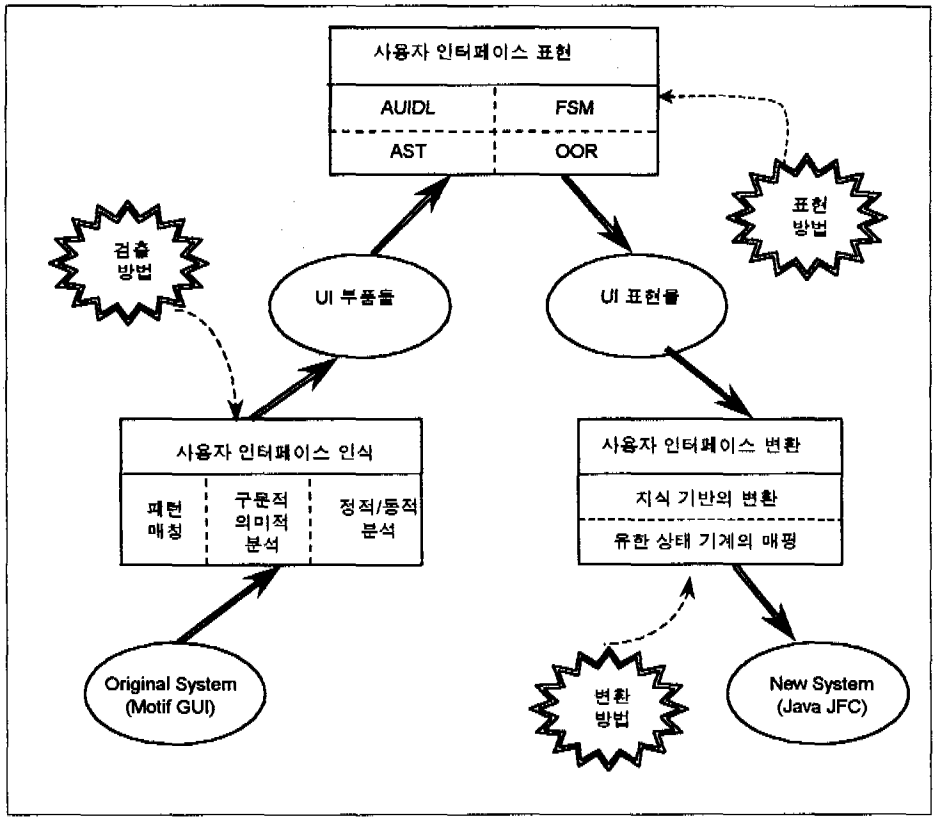

그림 4 Motif GUI를 Java JFC로 변환하는 사용자 인터페이스 이전 방법

4. 결 론

급속히 변화하고 있는 컴퓨팅 환경 특히 인터넷을 중심으로 발전하고 있는 현재의 상황은 WWW 응용 프로그램에 대한 많은 수요를 요구하고 있다. 이런 응용 프로그램들을 처음부터 새로 개발하기 보다는 기존의 프로그램들을 재공학하여 사용하는 것이 훨씬 효과적이다. 특히 사용자 인터페이스가 많은 비중을 차지하고 있는 대화식 프로그램들을 재공학하기 위해 중요한 것은 사용자 인터페이스 이전에 대한 것이다.

본 논문에서는 사용자 인터페이스 재공학 기술의 정의와 고려사항, 그리고 사용자 인터페이스 재공학 프로세스에 대해 기술하였다. 이 프로세스는 사용자 인터페이스를 인식하는 검출 단계, 이를 표현하고 저장하는 표현 단계, 그리고 새로운 환경에 맞게 변환하는 변환 단계로 구성된다. 또한 Unix 환경의 대표적인 그래픽 사용자 인터페이스인 Motif 를 WWW 어플리케이션의 대표 주자가 될 것으로 예측되는 Java JFC로의 변환 방법에 대하여 제안하였다.

제안한 방법을 통해 기존의 Motif 응용 프로그램들을 WWW 환경에서도 사용할 수 있는 기반을 구축할 수 있으므로, 이들을 다시 재작성해야 하는 노력을 덜 수 있고, 궁극적

으로는 기존의 Unix 환경에서 개발된 많은 X 응용 프로그램들을 WWW 하에서 사용 가능하게 될 것으로 기대된다.

향후에는 전체 응용 프로그램을 새로운 환경에 맞추어 쉽게 이전시키는 적응 유지보수(adaptative maintenance)에 대한 기술에 대한 연구가 진행되어야 할 것이다. 구체적으로 Motif 위젯셀이 아닌 다른 위젯셀을 사용한 X 응용 프로그램도 쉽게 이전시킬 수 있는 사용자 인터페이스 재공학에 대한 일반적인 도구를 개발하는 연구, 기존 언어와 Java 언어와의 언어 변환에 대한 연구를 추가로 진행하여, 사용자 인터페이스를 포함한 기존 시스템 전체 코드를 Java 언어로 이식가능하게 하여, WWW 상에서 운용되는 응용프로그램에 대한 개발 수요를 충족시켜줄 필요가 있을 것이다.

참 고 문 헌

[1] R.Arnold, Software Reengineering, IEEE Computer Society Press, Los Alamitos, 1993.

[2] J.Sutton, R.Sprague, "A Survey of Business Applications," Proc. of the American Institute for Decision Sciences 10th Annual Conference, Part II, Atlanta, GA, 1978.

[3] Georgia Istitute of Technology, Summary Report of the Meeting of the Reuse Issues Action Team(RIAT), http://columbia.ivv.nasa.gov:6600/RIAT/minutes/9701.

[4] M.Moore, R.Spencer, "Issues in User Interface Migration," Proc. of Software Engineering Research Forum, Orlando, Florida, Nov. 1993.

[5] I.Classen, K.Hennig, I.Mohr, M.Schulz, "CUI to GUI Migration: Static Analysis of Character-Based Panels," Proc. of 1st Euromicro Working Conference on Software Maintenance and Reengineering, IEEE Comp. Soc. Press, 1997, pp.144-149.

[6] E.Merlo, J.F.Girard, K.Kontogiannis, P.Panangaden, R.De Mori, "Reverse Engineering of User Interfaces," Proc. of Working Conf. on Reverse Enginerring, Baltimore, USA, IEEE Comp. Soc. Press, May 1993, pp.171-179.

[7] E.Merlo, J.F.Girard, L.Hendren, R.De Mori, "Multi-Valued Constant Propogation for the Reengineering of User Interfaces," Proc. of IEEE Conf. on Software Maintenance, Quebec, Canada, IEEE Comp. Soc. Press, 1993, pp.120-129.

[8] E.Merlo, P.Gagne, J.F.Girard, K.Kontogiannis, L.Hendren, P.Panangaden, R.De Mori, "Reengineering User Interfaces," IEEE Software, Vol. 12, No.1, Jan. 1995, pp.64-73.

[9] G.Antoniol, R.Fiutem, E.Merlo, P.Tonella, "Application and User Interface Migration from BASIC to Visual C++," Proc. of IEEE Conf. on Software Maintenance, Nice, France, IEEE Comp. Soc. Press, Oct. 1995, pp.76-85.

[10] L.V.Sickle, Z.Y.Liu, M.Ballantyne, "Recovering User Interface Specifications for Porting Transaction Processing Applications," Proc. of Workshop on Program Comprehension, Capri, Italy, IEEE Comp. Soc. Press, July 1993, pp.71-76.

[11] L.Csaba, "Experience with User Interface Reengineering Transferring DOS Panels to Windows," Proc. of 1st Euromicro Working Conference on Software Maintenance

and Reengineering, IEEE Comp. Soc. Press, 1997, pp.150-155.

[12] M.Moore, S.Rugaber, P.Seaver, "Knowledge-based User Interface Migration," Proc. of the IEEE Conference on Software Maintenance, Victoria, 1994, pp.72-79.

[13] M.Moore, R.Hobbs, S.Rugaber, "Reengineering Army Software to an Open Systems Environment," Reengineering Newsletter, IEEE TCSE, No.14, Spring 1997, pp.5-10.

[14] M.Moore, "Rule-Based Detection for Reverse Engineering User Interfaces," Proc. of Working Conf. on Reverse Enginerring, Calif, USA, IEEE Comp. Soc. Press, Nov. 1996, pp.42-48.

[15] J.Foley, W.Kim, S.Kovacevic, K.Murry, "UIDE - An Intelligent User Interface Design Environment," in Intelligient User Interfaces, edited by Sullivan & Tyler, ACM Press, 1991.

[16] D.Flanagan, X Toolkit Intrinsics Reference Manual, O'Reilly & Associations, Inc., 1992.

[17] P.M.Ferguson, D.Brennan, Motif Reference Manual, O'Reilly & Associations, Inc., 1993.