레거시 시스템의 사용자 인터페이스 컴포넌트화 프로세스

조영호[0] 최윤석 정기원*
숭실대학교 컴퓨터 학과, 숭실대학교 컴퓨터학부*
snatcher93@hanmail.net, yschoi@it.soongsil.ac.kr, chong@computing.soongsil.ac.kr

The Transition Process from User Interfaces of Legacy Systems to Components

Young-Ho Cho[0] Youn-Seok Choi Ki-Won Chong*
Dept. of Computing, Soongsil Univ. School of Computing, Soongsil Univ.

요 약

최근 객체지향 기법보다 효율적인 재사용성과 대체성을 지원하는 컴포넌트 기법에 대한 인식의 확대로 많은 업무용 시스템들에 대한 컴포넌트 기반 시스템으로의 재개발 또는 재공학 요구가 증가하고 있으며 이에 따라 레거시 시스템을 컴포넌트 기반 시스템으로 전환하기 위한 연구 또한 활발히 진행되고 있다.

본 논문에서는 레거시 시스템의 사용자 인터페이스를 컴포넌트화 하기 위한 전환 프로세스를 제시한다. 전환 프로세스는 레거시 시스템 사용자 인터페이스 구성 분석, 컴포넌트 유형 분석, 컴포넌트 전환 패턴 개발, 컴포넌트 설계 및 구현, 사용자 인터페이스 요구사항 테스트의 다섯 단계로 구성되어 있다. 레거시 시스템 사용자 인터페이스 구성 분석단계에서는 사용자 인터페이스의 화면구성과 내부로직을 분석하고 사용자 인터페이스의 컴포넌트화 단위를 결정하며 컴포넌트 유형 분석단계에서는 사용자 인터페이스를 구현할 컴포넌트 기술을 결정한다. 컴포넌트 전환 패턴 개발단계에서는 사용자 인터페이스를 컴포넌트화 시키기 위한 전환 패턴을 설계하며 컴포넌트 설계 및 구현 단계에서는 설계된 전환 패턴에 따라 사용자 인터페이스를 컴포넌트화한다. 사용자 인터페이스 요구사항 테스트단계에서는 개발된 컴포넌트가 레거시 시스템의 사용자 인터페이스에 대한 요구사항을 만족하는가를 확인한다. 제시한 프로세스를 적용한 레거시 시스템의 사용자 인터페이스를 컴포넌트화한 구현 사례를 통해 사용자 인터페이스와 관련된 코드의 재사용 방법을 제시한다.

1. 서 론

최근 객체지향 기법보다 효율적인 재사용성과 대체성을 지원하는 컴포넌트 기법에 대한 인식의 확대로 많은 업무용 시스템이 웹을 중심으로 한 컴포넌트 기반 시스템으로 개발되고 있다. 이로 인해 레거시 시스템을 컴포넌트 기반 시스템 환경으로 전환하기 위한 재개발 또는 재공학 연구 역시 활발히 진행되고 있다.

레거시 시스템의 사용자 인터페이스는 일반적으로 GUI 방식의 인터페이스를 채택하고 있다. 이러한 방식의 사용자 인터페이스를 웹 기반의 사용자 인터페이스 컴포넌트로 전환시키기 위해서는 웹 환경에 적합한 형태의 사용자 인터페이스로 재개발하는 것이 일반적이다. 그러나 사용자 인터페이스를 재개발하는 것보다는 기존 시스템의 사용자 인터페이스를 컴포넌트로 전환시키는 것이 시간과 비용 측면에서 더 효과적이다. 이런 방법은 사용자 인터페이스에 대한 사용자 요구사항을 자연스럽게 수용할 수 있으며, 레거시 시스템의 사용자 인터페이스가 그대로 유지되기 때문에 기존 사용자가 신규 시스템으로 전환하는데 도움을 줄 수 있을 것으로 기대된다.

이에 따라 본 논문에서는 레거시 시스템의 사용자 인터페이스를 컴포넌트화 시키기 위한 기법을 제시하였다.

2. 관련연구

2.1 RAD(Rapid Application Development) 모형에서의 사용자 인터페이스 개발

RAD는 매우 짧은 주기를 강조하는 소프트웨어 개발 프로세스로 컴포넌트를 기반으로 하는 구축 접근법을 사용하는 고속 적응 모형이다. RAD는 기존의 3세대 프로그래밍 언어 대신 4세대 프로그래밍 언어를 사용한다. 또한 기존의 프로그램 컴포넌트를 재사용하거나 재사용 가능한 컴포넌트 개발 시 자동화 도구를 사용해 소프트웨어 개발을 촉진시킨다.

RAD를 위한 자동화 도구의 대표적인 것이 사용자 인터페이스 개발 도구이다. 사용자 인터페이스 개발 방식에는 툴킷을 작성해 인터페이스에 필요한 함수를 저장한 후 필요 시 이를 호출하는 '사용자 인터페이스 툴킷' 방식과 설계자가 특수목적 언어를 사용해 인터페이스를 정의하는 '사용자 인터페이스 구문언어' 방식, 마우스를 이용해 스크린 상에 객체를 위치시킴으로써 인터페이스를 정의하는 '사용자 인터페이스 직접 조작' 방식, 윈도우와 같은 그래픽 툴을 사용해 사용자 인터페이스를 정의하는 '그래픽 사용자 인터페이스' 방식이 있다[1].

윈도우와 같이 GUI를 기반으로 하는 응용 프로그램의 경우 '그래픽 사용자 인터페이스' 방식의 자동화 도구의 지원을 통해 고속 개발을 지원하고 있다. 그러나 사용자 인터페이스를 컴포넌트화 할 경우 이에 대한 지원이 현재까지는 미흡하다.

2.2 사용자 인터페이스 컴포넌트

사용자 인터페이스는 응용 프로그램에서 커다란 부분을 차지함에도 불구하고 툴킷(위젯) 코드와 같이 제한된 범위 내에서만 재사용되고 있다[2]. 또한 툴킷의 배포 시 바이너리뿐만 아니라 소스 코드까지 함께 사용자에게 배포해야 한다는 단점이 있다.

이러한 단점을 극복하고 소프트웨어 개발비용과 기간을 단축하기 위해 사용자 인터페이스를 컴포넌트화하기 위한 다양한 종류의 기술이 발표되고 있다. 사용자 인터페이스 개발에 적용되고 있는 대표적인 컴포넌트 기술로는 마이크로소프트사의 ActiveX 컨트롤과, 썬 마이크로시스템사의 자바빈즈를 들 수 있다.

자바빈즈는 객체지향 프로그래밍 인터페이스로서 플랫폼의 네트워크 내에 적용될 수 있는 재사용 가능한 애플리케이션이나 프로그램 빌딩 블록과 같은 컴포넌트를 구축할 수 있게 한다. 자바 애플릿처럼, 자바빈즈 컴포넌트들(일명 빈즈)도 사용자 또는 브라우저 특성에 맞게 페이지 내용을 변경하는 등, 웹페이지에 상호작용적인 기능을 부여하는데 사용될 수 있다.

ActiveX 컨트롤은 컴포넌트를 일반적인 컨트롤처럼 보이고 행동하도록 만들기 위해 여러 가지 표준 인터페이스를 구현한 COM 오브젝트이다. ActiveX 컨트롤은 두 가지 목적으로 사용될 수 있다. 첫 번째 응용 프로그램 개발자가 자신의 프로그램을 개선하기 위해 ActiveX 컨트롤을 사용할 수 있으며, 두 번째 웹 개발자가 자신이 개발한 웹 컨텐츠가 사용자와 상호작용 할 수 있도록 만들 때 ActiveX 컨트롤을 사용할 수 있다[3].

이와 더불어 웹 상에서 사용자와의 상호작용을 돕기 위해 사용되는 자바 애플릿 역시 넓은 의미에서의 사용자 인터페이스 컴포넌트라고 할 수 있다.

3. 레거시 시스템의 사용자 인터페이스 컴포넌트화 프로세스

레거시 시스템을 컴포넌트로 전환하는 경우 기존에 작성되어 있는 사용자 인터페이스 관련 코드 및 리소스를 재사용하기 보다는 새로 작

성하는 것이 일반적이다. 현재까지 'GUI 사용자 인터페이스' 방식의 자동화 도구를 지원하고 있는 개발툴의 경우 사용자 인터페이스 컴포넌트 개발을 위한 자동화 도구의 지원이 미흡할 뿐만 아니라 레거시 시스템의 개발 환경과 컴포넌트 시스템의 개발환경간의 연계성이 부족하다. 따라서 재개발을 통해 레거시 시스템의 사용자 인터페이스를 컴포넌트화하는 경우 동일한 사용자 인터페이스 개발을 위해 비용과 시간이 중복 투자되며 서로 다른 개발환경에서 동일한 기능과 화면구성을 제공하기 위한 방법을 개발해야 하는 부담을 감수해야 한다.

본 논문에서는 레거시 시스템의 사용자 인터페이스를 컴포넌트화 하여 개발비용과 시간을 절약하고 코드 및 리소스에 대한 재사용성을 향상시킬 수 있는 프로세스를 제시하고자 한다. <그림 1>은 이런 프로세스의 단계를 나타낸다.

<그림 1> 레거시 시스템의 사용자 인터페이스 컴포넌트화 프로세스

레거시 시스템 사용자 인터페이스 구성 분석단계는 레거시 시스템의 사용자 인터페이스를 웹 환경에서 사용할 수 있도록 컴포넌트화하기 위해 사용자 인터페이스 화면구성 및 내부로직을 분석하고 어떤 단위로 이를 컴포넌트화 할 것인지를 결정한다. 이때 분석의 기본 자료로 활용하기 위해 역공학을 수행하여 이를 통해 얻어진 결과물을 사용하거나 레거시 시스템 개발 시의 산출물을 이용할 수 있다. 분석과정을 통해 사용자 인터페이스에 새로이 추가될 요구사항이나 누락된 기능을 식별한다.

레거시 시스템 사용자 인터페이스 구성 분석단계에서는 레거시 시스템의 사용자 인터페이스를 웹 환경에서 사용할 수 있도록 컴포넌트화하기 위해 사용자 인터페이스의 화면구성과 내부로직을 분석한다. 이때 분석의 기본 자료로 활용하기 위해 역공학을 수행하여 이를 통해 얻어진 결과물을 사용하거나 레거시 시스템 개발 시의 산출물을 이용할 수 있다. 분석과정을 통해 사용자 인터페이스에 새로이 추가될 요구사항이나 누락된 기능을 식별한다.

사용자 인터페이스 구성 분석 시 사용자 인터페이스를 컴포넌트화 시키는 단위에 대한 고려가 필요하다. 일반적인 경우 사용자 인터페이스의 컴포넌트화 단위는 컨트롤 각각이 되지만 사용자 요구사항에 따라 컨트롤 그룹이나 사용자 인터페이스 전체를 하나의 컴포넌트화할 수도 있다.

컴포넌트 유형 분석단계에서는 전 단계에서 도출한 레거시 시스템의 사용자 인터페이스 구조와 컴포넌트화 단위를 바탕으로 해당 인터페이스를 어떤 컴포넌트 기술로 구현할 것인지를 결정한다. 이때 기존 사용자 인터페이스에 관련된 소스코드나 알고리즘의 재사용을 고려해야 하며 컴포넌트로 전환 시 기존 사용자 인터페이스의 기능을 최대한 반영할 수 있는 컴포넌트 구현 기술을 채택해야 한다. 사용자 인터페이스의 컴포넌트 구현 기술에는 자바빈즈나 ActiveX 등이 있으며 운용 플랫폼, 개발 언어, 특정 기술의 지원 여부와 각 컴포넌트 기술에 대한 특징 및 장단점을 파악해 가장 적절한 기술을 도입하는 것이 필요하다.

컴포넌트 전환 패턴 개발단계에서는 사용자 인터페이스가 가져야 할 컴포넌트 인터페이스를 식별하고, 이벤트 처리방식을 고려해 레거시 시스템의 사용자 인터페이스를 컴포넌트화 시키기 위한 전환 패턴을 설계한다.

하나의 언어나 툴킷을 사용하여 개발된 응용 프로그램의 경우 사용자 인터페이스 부분은 비슷한 구조와 내부 로직을 가지며 일정한 규칙에 따라 사용자의 요구에 응답한다. 따라서 사용자 인터페이스의 구조와 로직에 대한 공통점과 차이점을 식별하여 이에 따라 전환 패턴을 설계하면 다량의 사용자 인터페이스를 동일한 패턴에 따라 손쉽게 컴포넌트화 시킬 수 있으며 개발에 소요되는 시간과 비용을 줄일 수 있다. 또한 기존 사용자 인터페이스와 관련된 소스 코드 및 리소스의 재사용성을 높일 수 있다.

설계된 패턴은 레거시 시스템의 사용자 인터페이스 관련 코드 및 리소스를 재사용할 수 있는 프레임워크를 제공해야 하며 개발된 컴포넌트를 레거시 시스템에 부착시킬 수 있는 메커니즘을 제공해야 한다.

컴포넌트 설계 및 구현단계에서는 개발된 전환 패턴을 바탕으로 실제 레거시 시스템의 사용자 인터페이스를 컴포넌트화된 사용자 인터페이스로 설계, 구현한다.

사용자 인터페이스를 컴포넌트화할 수 있는 패턴을 개발했다고 해서 기존의 코드 및 리소스를 100% 재사용할 수 있는 것은 아니다. 컴포넌트의 경우 계약관계를 나타내는 인터페이스를 통해 클라이언트에게 서비스를 제공하기 때문에 레거시 시스템과 상이한 구조를 가질 수밖에 없으며 코드 역시 컴포넌트의 특성을 반영하기 위해 수정이 불가피하다. 또한 설계된 전환 패턴은 단순히 재사용을 고려한 표준적인 매핑 방법만을 나타낼 뿐 사용자 인터페이스의 소스 코드에 대한 수정 절차는 포함하고 있지 않다.

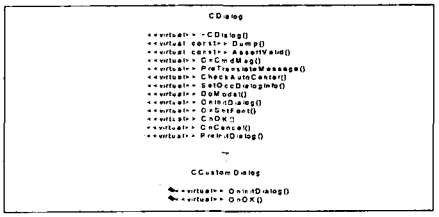
따라서 기존의 코드에서 수정 및 재작성 되어야 하는 부분을 식별하고 이를 설계된 전환 패턴에 적합하게 수정할 수 있도록 하기 위한 코드 수정 절차를 정의한다. 컴포넌트 개발자는 정의된 코드 수정 절차 및 전환 패턴을 적용하여 레거시 시스템의 각 사용자 인터페이스를 효율적으로 컴포넌트화 시킬 수 있다.

사용자 인터페이스 요구사항 테스트단계에서는 개발된 컴포넌트가 레거시 시스템의 사용자 인터페이스에 대한 요구사항을 만족하는가를 확인한다. 동일한 기능과 구조를 제공하는지, 사용자 입력에 대해 동일하게 응답하는지에 대한 기능적 요구사항 뿐만 아니라 응답 속도, 보안, 처리 속도와 같은 비기능적 요구사항 역시 만족하는지를 테스트한다.

4. 적용 사례

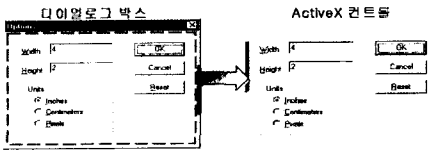
본 논문에서는 3장에서 제시한 사용자 인터페이스의 컴포넌트화 방법을 적용하기 위해 이미지 처리 프로그램의 사용자 인터페이스를 컴포넌트화 하였다. 대상 프로그램은 기존의 이미지를 사용하거나 외부 기기를 통해 이미지를 획득하고 획득된 이미지의 변환 및 분석, 통계처리를 수행하는 프로그램이다. 윈도우즈 플랫폼에서 지원되는 클래스 라이브러리인 MFC를 사용해 개발되었으며 다이얼로그 기반의 사용자 인터페이스를 중심으로 구성되어 있다.

레거시 시스템 사용자 인터페이스 구성 분석단계에서 역공학을 수행한 결과 이미지 처리 프로그램은 다이얼로그 박스를 기반으로 하고 있었으며 MFC의 CDialog 클래스의 파생 클래스를 통해 다이얼로그를 생성하는 구조를 가지고 있었다. <그림 2>는 역공학을 통해 분석한 사용자 인터페이스에 대한 클래스 다이어그램을 나타낸 것이다.



<그림 2> 사용자 인터페이스의 클래스 다이어그램

기존의 다이얼로그 박스를 구성하고 있는 전체 컨트롤 그룹이 컴포넌트로 도출되었다. 다이얼로그 박스 내의 전체 컨트롤을 컴포넌트화하는 이유는 응용 프로그램에서 사용하는 사용자 인터페이스를 동적으로 임베딩하기 위해서이며 동일한 사용자 인터페이스를 웹이나 일반 응용 프로그램 모두에서 사용하기 위해서이다. <그림 3>은 레거시 시스템 사용자 인터페이스 구성 분석 단계에서 도출된 사용자 인터페이스의 컴포넌트화 단위를 나타낸다.

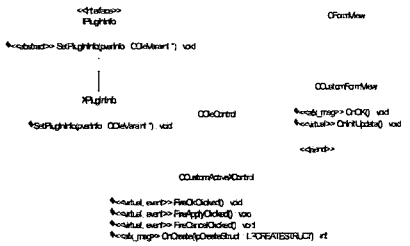


<그림 3> 사용자 인터페이스의 컴포넌트화 단위

컴포넌트를 실행시간에 임베딩하기 위해서는 컨테이너의 역할을 수행하는 다이얼로그 박스를 동적으로 생성할 수 있어야 하며, 생성된 다이얼로그 박스 역시 임베딩할 컴포넌트를 동적으로 선택할 수 있어야 한다. 이를 위해 마이크로소프트사의 COM을 사용하였으며, 사용자

인터페이스를 하나의 컨트롤 단위로 배포하기 위해 COM 기반기술인 ActiveX 컨트롤을 사용하였다.

다음으로 컴포넌트 전환패턴 개발단계를 통해 <그림 2>의 다이얼로그 박스를 ActiveX 컨트롤로 변환할 수 있는 패턴을 개발하였다. 다이얼로그 박스의 컨트롤 그룹을 컴포넌트화 할 경우 가장 큰 문제점은 기존에 개발되어 있는 다이얼로그 템플릿과 소스 코드의 재사용이 불가능하다는 점이다. 이를 위해 다이얼로그를 담당하는 클래스를 화면을 담당하는 새로운 클래스로 변환하고, 이를 ActiveX 컨트롤의 내부에 삽입하는 방법을 사용하였다. 예를 들어 기존의 CDialog 파생클래스가 갖는 프레임 특성 또는 컨트롤 배치와 같은 리소스 사용을 가능하게 하기 위해 CDialog를 CFormView 클래스로 변환하였다[4]. 이처럼 CFormView를 사용해 사용자 인터페이스를 컴포넌트화 할 경우 다이얼로그 템플릿은 완벽하게 재사용 가능하며 관련된 소스코드 역시 약간의 수정을 통해 재사용 가능하였다. 다음 <그림 4>는 CFormView를 사용해 다이얼로그 박스 기반의 사용자 인터페이스를 ActiveX 컨트롤로 변환하기 위해 설계한 패턴이다.



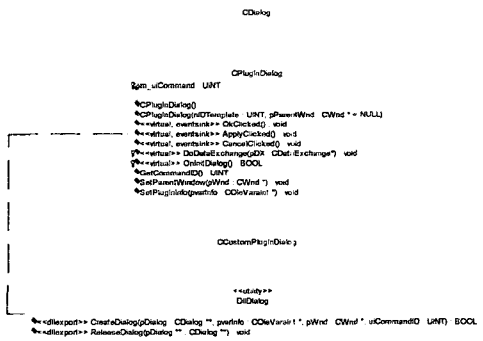
<그림 4> ActiveX 컨트롤로의 변환을 위한 패턴

개발된 패턴은 일반적인 ActiveX 컨트롤 개발 패턴과는 달리 소스코드의 재사용을 위해 컨트롤 내부에 CFormView 파생클래스를 포함하였다. 또한 레거시 시스템의 사용자 인터페이스와 동일한 기능을 지원하기 위해 버튼 클릭과 같은 사용자 이벤트를 통지할 수 있는 메소드를 추가하였다.

대부분의 응용 프로그램은 사용자 인터페이스를 화면에 출력하기 전에 화면을 구성하고 있는 각 항목에 대해 적절한 초기화를 수행한다. 그러나 ActiveX 컨트롤과 같은 컴포넌트의 경우 사용자 인터페이스의 직접적인 초기화는 불가능하며 컴포넌트의 인터페이스를 통해서만 이런 기능을 수행할 수 있다. 개발된 패턴에서는 이를 위해 IPlugInInfo 인터페이스를 정의해 클라이언트로 하여금 다이얼로그 박스의 초기화를 수행할 수 있도록 하였다.

<그림 4>의 패턴을 사용해 개발된 ActiveX컨트롤의 경우 웹 페이지에는 그대로 임베딩 시킬 수 있으나 응용 프로그램일 경우에는 임베딩을 위한 별도의 메커니즘을 필요로 한다. 본 논문에서는 이를 위해 응용 프로그램과 ActiveX 컨트롤 사이에 동적 라이브러리를 두어 동적으로 다이얼로그 박스를 생성할 수 있도록 하였으며, 생성된 다이얼로그 박스가 ActiveX 컨트롤을 동적으로 내부에 임베딩하도록 하였다.

<그림 5>는 ActiveX 컨트롤을 동적으로 다이얼로그 내에 임베딩하기 위해 사용되는 패턴을 나타낸 것이다.



<그림 5> ActiveX 컨트롤을 임베딩하기 위한 패턴

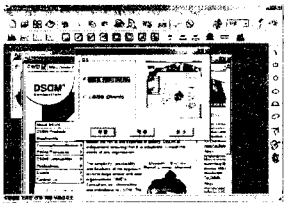
컴포넌트 설계 및 구현 단계에서는 <그림 4>와 <그림 5>의 패턴을 적용해 사용자 인터페이스를 컴포넌트화 하였다. <표 1>은 실제로 이미지 처리 프로그램의 사용자 인터페이스를 MFC를 이용해 컴포넌트화 할 때 적용한 수정 절차의 예를 나타낸 것이다. 이런 수정 절차의 정의를 통해 사용자 인터페이스의 소스 코드 대부분을 재사용할 수 있다.

<표 1> 다이얼로그 박스를 컴포넌트화 하기 위한 코드 수정 절차의 예

단 계	절 차
1	기존에 작성된 다이얼로그 템플릿을 리소스 스크립트에 포함한다
2	해당 다이얼로그 박스에 관련된 CDialog 파생 클래스를 CFormView 파생 클래스로 변경한다
3	CDialog의 OnInitDialog()를 OnInitialUpdate()로 변환한다.
4	IDOK에 대한 버튼 클릭 이벤트를 메시지 맵에 추가한 후 OnOK()를 핸들러로 설정한다.
5	가상함수 OnOK()를 이벤트 핸들러로 변경한다.

사용자 인터페이스 요구사항 테스트 단계에서는 개발된 ActiveX 컨트롤이 레거시 시스템의 사용자 인터페이스에 대한 요구사항을 만족하는지 확인하였다. 컴포넌트의 테스트를 위해 프로그램의 메인 부분은 이미지를 관리할 수 있는 기능과 위에서 제시한 패턴을 수용할 수 있는 구조로 간략화 하였으며 기존의 처리 프로그램과 테스트 프로그램을 동시에 실행시켜 동일한 결과를 보이는 지를 검사하였다.

<그림 6>은 기존 시스템의 사용자 인터페이스를 컴포넌트화 시켜 구현한 이미지 처리 프로그램의 실행화면이다.



<그림 6> 이미지 처리 프로그램 실행 화면

5. 결 론

본 논문에서는 레거시 시스템의 사용자 인터페이스를 컴포넌트화 하기 위한 전환 프로세스를 제시하였다. 전환 프로세스는 레거시 시스템 사용자 인터페이스 구성 분석, 컴포넌트 유형 분석, 컴포넌트 전환 패턴 개발, 컴포넌트 설계 및 구현, 사용자 인터페이스 요구사항 테스트의 다섯 단계로 구성되어 있다. 이를 적용하여 이미지 처리 프로그램의 사용자 인터페이스를 컴포넌트화 하였으며 다이얼로그 박스를 구성하고 있는 전체 컨트롤 그룹을 컴포넌트의 단위로 도출하였다. 기존의 사용자 인터페이스 관련 코드의 재사용성을 높이기 위해 컴포넌트 변환 및 이를 임베딩하기 위한 패턴을 개발하였으며 기존의 프로그램의 요구사항을 만족하는지를 테스트하였다.

본 논문에서 제시한 프로세스를 적용한 결과 레거시 시스템의 사용자 인터페이스에 관련된 코드를 재사용할 수 있었으며 생성된 사용자 인터페이스를 동적으로 응용 프로그램에 임베딩시킬 수 있었다.

향후에는 모든 컴포넌트 기술에 공통적으로 사용될 수 있는 변환 패턴 및 재사용 기법에 관한 연구가 진행되어야 할 것이다. 이를 위해 다양한 컴포넌트 기술을 적용해 사용자 인터페이스를 컴포넌트화 한 후 각 변환 패턴의 공통점 및 차이점 식별에 관한 연구가 선행되어야 할 것이다.

6. 참고 문헌

[1] 홍찬기, "GUI의 재사용 컴포넌트 분류에 관한 연구", 관대논문집, No.1, pp 25-43, 1996.

[2] Richard N. Taylor, et al. "A Component-and Message-Based Architectural Style for GUI Software", IEEE transaction on Software Engineering, VOL 22, No.6, June 1996.

[3] Jeff Prosise, "Programming Windows with MFC, Second Edition", Microsoft Press, 1999.

[4] David J. Kruglinski, "Inside Visual C++ 5.0, Fourth Edition", Microsoft Press, 1997.