韓國 컴퓨터情報學會 論文誌
第10卷 第1號, 2005. 3.

2005-3-1-1-15

시각 프로그램을 위한 공동 개발 환경

조 용 윤*, 유 재 우**

A Co-Development Environment for Visual Programs

Yong-Yoon Cho*, Chae-Woo Yoo**

요 약

시각 프로그램은 GUI(graphic user interface)모듈과 프로세스 로직(logic) 모듈로 구성된다. 시각 프로그램의 빠른 개발을 위한 대부분의 RAD(rapid application development)툴 들은 큰 규모의 시각 프로그램 개발에 있어 다중 개발자 참여를 지원하지 않는다. 또한 기존의 공동 작업 도구는 우선권 및 병행성(concurrency) 제어에 있어 유연성이 부족한 단점이 있다. 본 논문은 시각 프로그램 개발에 있어 개발자간 즉각적인 의견 교환을 통한 분업 개발을 지원할 수 있는 공동 개발 환경을 제안하다. 제안하는 공동 개발 환경은 개발자간 다양한 개발 정보 교환과 의견 수렴을 위한 통신 창을 제공하며 공동 작업 공간에서의 개발 공유 객체에 대한 접근 우선권 및 병행성 제어를 위한 실시간 작업 제어기를 포함한다. 따라서 개발자들은 공동 개발 단계에서 충분하고 즉각적인 의견 수렴이 가능하고 공동 개발 과정에 대한 일관성과 신뢰성을 유지할 수 있다. 특히, 제안하는 공동 개발 환경은 대규모 시각 프로그램 개발에 있어 단일 개발자의 개발 부담을 줄이고 개발자의 개발 능력과 특성에 따라 분산 형태의 공동 개발 참여를 유도하여 관련 시각 프로그램 개발 속도와 효율을 높일 수 있을 것으로 기대된다.

Abstract

Visual programs consist of user interfaces and process logics. Because most visual IDEs(Integrated Development Environment) are utilized by only one developer, many developers can't cooperate simultaneously in one large-scaled visual program to promote the development efficiency. This paper suggests a co-development environment for visual programs through which developers can cooperate with each other in the type of distribution. Developers can maintain the coincidence and the confidence of cooperation through a communication window and a job controller to process the priority of the events that developers make in co-development work. Especially in large-scaled visual program, the suggested environment reduces the burden that developer takes about the heavy development work, and many developers can divide and take the complex and large program modules according to their ability. We hope that the suggested environment can improve the development productivity and effectivity because it can reduce the time and effort for developing user interfaces in visual programs.

▸ Keyword : Visual program, Co-development, User interface, Instance communication,

• 제1저자 : 조용윤
• 접수일 : 2005.01.28, 심사완료일 : 2005.03.05
*숭실대학교 컴퓨터학부 박사과정 ** 숭실대학교 컴퓨터학부 교수

Ⅰ. 서 론

대부분의 응용프로그램 개발 환경은 단일 설계자에게 프로그램 모듈과 모든 인터페이스 설계의 책임과 권한을 부여하기 때문에, 응용 프로그램 설계에 대한 모든 생각(idea)이 단일 설계자에게 집중된다. 때문에, 단일 개발자는 프로그램 요소(component)의 생성과 프로그램 로직(logic)의 정의 및 추가 코드 작성 등의 개발과정을 혼자 처리해야만 한다. 현재의 응용프로그램은 그 복잡성과 규모가 점점 증가하고 있고, 그에 따른 사용자들의 사용자 인터페이스에 대한 요구도 나날이 복잡해지고 있다[1][2]. 따라서 단일 개발자가 규모가 크고 복잡한 시각 프로그램의 전체 설계와 개발을 모두 담당해야 한다는 것은 부담이 아닐 수 없다. 또한 단일 개발자가 광범위한 설계에 대해 일관성을 유지하면서 전체적인 프로그램 개발 계획을 정확하게 숙지해야 한다는 사실은 실제로 어려운 일이다. 특히, 설계의 변경이 발생했을 경우, 단일 개발자는 프로그램의 모든 부분을 검색하여 변경될 부분을 파악하고 처리해야 하기 때문에 개발 시간의 증가를 가져올 수 있다.

이러한 모든 단점은 기존 시각 프로그램 개발 환경이 여러 개발자가 공동으로 개발에 참여할 수 있는 기능을 제공하지 않고, 단일 개발자 위주의 개발 기능을 제공하고 있기 때문이다. 본 논문은 여러 개발자가 협업을 통해 시각 프로그램 개발에 공동 참여할 수 있는 새로운 시각 프로그램을 위한 공동 개발 환경을 제안한다. 특히, 큰 규모의 시각 프로그램 개발에 있어서, 개발자는 개발 책임과 작업 관리부담을 줄이고, 빠른 시간 내에 더욱 간단하고 편리한 방법으로 시각 프로그램을 설계 할 수 있을 것이다. 프로그램 개발을 위한 공동 개발 환경은 TCP/IP 프로토콜을 이용해 클라이언트/서버 구조를 갖는 자바용 시각 프로그램 개발 툴 이다. 제안된 시스템은 프로그램 개발자간 협업을 위해 의견을 나눌 수 있는 통신 창을 제공하고 서로 공동의 작업창을 통해 협업할 수 있도록 개발자간 작업 제어 모듈을 제공 하는 공동 개발 환경이다.

Ⅱ. 관련 연구

2.1 시각 프로그램 개발 환경

프로그램 개발 초기 텍스트 기반 사용자 인터페이스는 GUI(Graphic User Interface) 형태의 더욱 편리하고 다양한 인터페이스로 발전하였으며, 시각 프로그램 개발 속도와 편리성을 제공하기 위한 시각 프로그램 개발환경이 개발되었다[1][2][3].

2.2 단일 개발자 환경

대표적인 단일 개발자 환경은 Free Builder, DRuid, X-Designer, Delphi, Visual Basic 등이 있다. 이 중에서 Free Builder는 자바 언어개발을 위한 통합적인 개발 환경으로, 여러 개발자들이 개방 구조(Open Architechure)형식으로 개발 보급한 소프트웨어이다. 현재 웹 환경에서 각광받고 있는 자바 언어를 개발하는데 있어 개발하고자 하는 응용 프로그램에 대한 전체적인 표현 부분과 동작 부분에 대한 사용자 인터페이스를 설계할 수 있는 여러 시스템을 갖추고 있다.

2.3 공동 개발자 환경

CVS 환경[3](concurrent version system)은 공동의 저장소(repository)를 두고 네트워크에 접속해 공동으로 프로그램을 개발하기 위한 방법이다. CVS 환경은 개발자들이 자신만의 프로젝트 폴더를 만들고 CVS 시스템이 제공하는 일관성(concurrency) 기능을 이용하여 프로그램을 공동으로 개발한다. 하지만, CVS를 이용한 공동 개발은 네트워크를 통해 파일 버전을 관리하며, 다른 개발자의 개발 파일을 자신의 컴퓨터에 복사해 개발하는 과정을 갖는다.

객체 기반의 GroupDraw 시스템[4]은 매킨토시 시스템에서 작동되도록 만들어졌으며, 완벽한 복제 구조로 되어있다. GroupDraw는 다른 참여자의 존재감(awareness)과 몸짓(gesture)을 표현하기 위해 여러 개의 커서를 이용하는 방법을 사용한다. GroupDraw는 사적 공간을 제공하기 위해 두 가지 방법을 제공한다. 첫 번째 방법은 스크롤을 제공하는 방법이다. 각 참여자는 스크롤을 이용해서 다른 장

소에서 사적인 작업을 한 뒤에 이 이미지를 공유 공간(public space)에 이동시킬 수 있다. 두 번째는 각 객체에 "커플링 상태(coupling status)"를 제공하는 방법이다. 따라서 한 사용자가 객체를 사용하고 있으면 다른 사용자는 즉시 이 상태를 알 수 있다.

CoDraw[5]는 객체 기반의 시스템으로서, CoDraw 시스템에서 제공하는 "자율 객체 (autonomous object)"와 클라이언트, 서버는 필요에 따라 병행성 제어 정책 및 시스템 구조를 유연하게 변경할 수 있다. CoDraw 시스템은 공동 작업이 가능한 드로잉 도구로서 객체 단위로 그림을 생성하거나 조작할 수 있다. 객체 기반의 시스템은 비트맵 기반의 시스템에 비해 보다 유연하고 편리한 기능들을 제공한다.

III. 프로그램 공동 개발 시스템

본 논문에서 제시하는 새로운 공동 개발 환경은 자바(java) 언어로 구현된 응용 프로그램 공동 개발 환경이다.

3.1 시스템

본 논문에서 제안하는 공동 개발 환경의 전체적인 시스템 모델은 (그림 1) 과 같다.

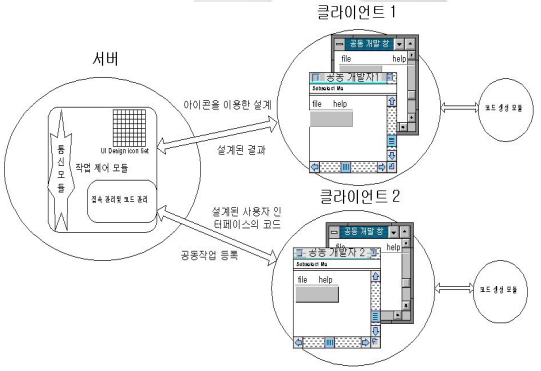

그림 1 프로그램 공동 개발 지원 시스템
Fig. 1 A program co-development system

제안되는 응용 프로그램 공동 개발 환경은 네트워크 상으로 여러 사용자가 동시에 접근 가능한 공동 작업 창과 각 개발자 창을 제공하는 개발 모듈과 공동 개발의 신뢰성 있는 작업진행과 개발자간 동기 적인(synchronous)통신을 보장하기 위한 작업 제어 모듈, 그리고 개발된 사용자 인터페이스에 대한 코드 생성 모듈을 포함하는 통합 시스템이다.

서버에서는 여러 개발자간의 공동 개발을 위한 접근 제어와 작업 제어가 수행되며, 클라이언트에서는 각 개발자가 발생한 이벤트와 통신 메시지를 서버에 전달하고 생성된 요소에 대한 자바 원시코드 생성이 수행된다. 서버는 스레드로 생성된 네트워크 관리 데몬들로 구성된다. 네트워크 관리 데몬들은 연결 전용 데몬과 클라이언트로부터 받은 메시지를 모든 참여자에게 브로드 케스팅하기 위한 메시지 데몬, 그리고 작업의 일관성(concurrency)유지를 위한 컨커론시(concurrency) 데몬으로 구성된다.

서버는 개발 모듈을 통해 공동 작업 창과 개발자 개인 창을 클라이언트의 공동 개발자들에게 제공한다. 개발 모듈은 실질적인 인터페이스를 공동 작업하기 위한 부분으로 사용자 인터페이스 개발을 위한 아이콘(icon)형태의 도구(tools)를 제공한다. 이러한 도구는 자바(java)에서 제공하는 사용자 인터페이스 요소(component)를 나타내며, 자바(java) 클래스로 구현된다. 따라서 개발 모듈은 목표 사용자 인터페이스를 작성하기 위해 자바로 구현된 개발 도구 클래스들로 구성되어 있으며, 자체는 자바로 구현된 사용자 인터페이스 개발 툴의 형태를 갖는다.

작업 제어 모듈은 사용자 인터페이스 공동 개발의 신뢰성과 보안성을 보장하기 위한 모듈이다. 제어 모듈은 접근 제어 모듈, 작업 창 생성 모듈 그리고 통신 모듈로 구성된다. 접근 제어 모듈은 공동 개발자의 초기 접근을 제어하기 위한 것이며, 우선권 제어 모듈은 개발자간 동일 요소에 대한 접근 순서와 우선권 결정 및 관리를 담당한다. 통신 모듈을 통해 개발자간 발생한 이벤트(event)는 모든 참여자에게 공유되기 위해 브로드 케스팅 된다[6][7][8][9][10].

코드 생성기는 개발자가 설계한 사용자 인터페이스의 독립적인 코드 생성으로 인한 프로그램의 재 이용성을 높인다. 제어 모듈은 새로운 시스템에서 가장 중요한 부분을 차지하는 모듈이다. 먼저 접근 제어 모듈은 공동 개발자가 초기 공동 작업장에 참여하려고 할 때, 특정 참여자의 참여 여부를 검증하는 기능을 가진다. 공동 작업 창과 개발자 개인 창 생성 모듈은 참여한 공동 개발자에게 공동 작업 창과 개인 창을 제공하며, 각 개발자가 생성하는 작업의 결과를 볼 수 있다. 마지막으로 통신 모듈은 개발자간 통신을 관리하고 제어하는 역할을 포함한다.

다음은 본 논문에서 제안하는 공동 개발 환경의 각 주요

부분에 대해 살펴보도록 한다.

3.2 제어모듈

제어모듈은 공동 개발에 참여하려는 개발자의 접근을 제어하고, 공동 개발 환경에서 이루어지는 모든 작업과정을 제어하는 부분이다. 제어모듈은 각 개발자의 설계과정에서 이루어지는 모든 입력에 대한 출력을 공동 개발 환경 화면에 직접 조작(direct-manipulation)방식을 통해 화면에 출력한다. 또한 각 개발자가 프로그램 개발 중 발생하는 의견을 전달할 수 있도록 통신 모듈을 제공한다. 다음은 주요 서브 모듈(module)들에 대해서 자세히 알아본다.

3.2.1 접근 제어 모듈

작업 제어 모듈은 공동 개발 참여자의 참여 접근을 제어하기 위한 연결(Connection) 클래스를 제공한다. 개발자의 개발참여 제어를 위한 서버/클라이언트 기반의 접근 제어 모델은 (그림 2)와 같다.

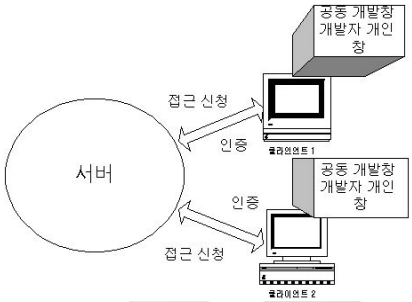

그림 2 클라이언트/서버 기반의 접근제어
Fig. 2 Aceess Control based on client/server

연결 클래스 내에는 로그인(login)정보를 얻기 위한 로그인(login) 클래스와 로그인한 개발자간 협의를 지원하는 통신 클래스가 있다.

3.2.2 공동 작업 창과 개발자 통신 창 제어 모듈

공동 작업 창과 개발자 통신 창 제어 모듈은 제어 모듈 중에서 가장 큰 부분을 차지하며, 여러 개발자간 공동 작업 창에서의 작업 대상 요소의 동시 접근에 대한 우선 접근 권한 부여에 관련된 제어를 수행한다. 또한, 공동 작업 창과 개인 작업 창에서 이루어지는 작업에 대해 일관성(concurrency)을 유지하기 위해 일관성 유지 모듈과 유기적으로 동작한다. 로그 인에 성공한 개발자는 공동 작업 창과 개인 창을 통해 공동 개발에 참여할 수 있다. 이때 새로 참여하는 개발자는 이전에 참여하고 있던 개발자들에게 보여지게 된다.

3.2.3 통신모듈

통신 모듈은 초기 공동 개발 참여자의 등록정보 전달을 포함하는 개발자간 정보의 전달 및 통신을 위한 클래스를 포함한다. 공동 개발환경은 기본적으로 TCP/IP 소켓 프로그램을 이용한 client/server구조이다. 서버는 클라이언트 즉, 공동 개발자에게 연결을 위한 연결 클래스내에 포함되어 있는 데몬을 제공하고, 적절한 메시지 정보를 생성하고 관리한다. 서버는 모든 공동 작업이 진행되는 실질적인 호스트이며, 개발자에게 개발 툴 셋을 제공한다. 클라이언트 호스트는 해당 호스트에 있는 개발자에게 서버 호스트로의 연결을 위한 데몬을 제공하고, 연결이 이루어지면 공동 개발 환경 초기 화면과 자신의 개발자 개인 창을 보여주는 역할을 수행하는 화면 조정 데몬을 제공한다. 서버는 클라이언트로부터의 연결을 위해 연결 데몬을 스레드로 생성하고, 이후 모든 제어를 실시한다. 클라이언트는 지역 호스트에서 공동 개발을 제어하는 서버에 연결하고 자신의 호스트에 제공되는 공동 개발 창과 개발자 개인 창에서 개발 툴 킷을 사용해 공동 개발을 시작한다.

3.2.4 메시지

공동 개발환경에서 발생되는 메시지는 공동 개발자간 공동 개발중에 서버와 지역 호스트간에 발생되는 이벤트 메시지와 공동 개발자간 협의를 위한 통신 메시지가 있다. 다음 (그림 3)은 통신 모듈을 통해 개발자간 발생된 메시지가 브로드 케스팅 되어지는 모습이다.

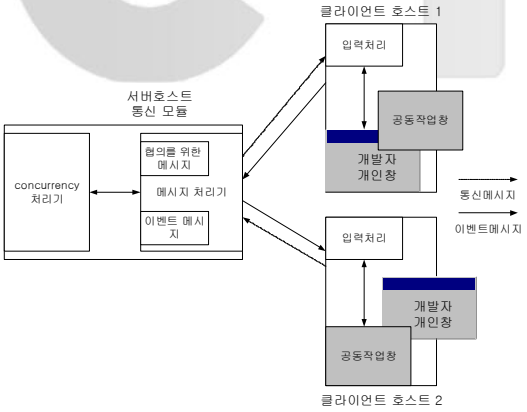

그림 3 통신 메시지 브로드케스팅 개념도
Fig. 3 Communication Broadcast Diagram

통신메시지의 형태에 따라 메시지를 처리하는 통신 모듈이 서로 구별되며, 각 메시지는 메시지 처리기에 의해 개발자의 개인 통신 창과 공동 개발 창에 브로드 케스팅 된다.

메시지 형태	개발자 고유 id	메시지 내용

공동 개발자간 통신 메시지는 스트링 형태 메시지 형태와 개발자 고유 id와 메시지 내용으로 구성된다. 메시지 형태는 메시지가 공동 개발자간 통신을 위한 메시지인 것을 나타내 주기 위한 것이다. 개발자 고유 id는 서버와 클라이언트를 연결하기 위한 것이다. 메시지 내용은 실제 메시지 내용이다.

메시지 형태	개발자 고유 id	프로젝트 이름	요소 종류	요소 속성

각 공동 참여자간 발생되는 이벤트 메시지 구조는 메시지 형태, 개발자 고유 id, 프로젝트 이름, 이벤트 요소 종류, 그리고 요소 속성으로 구성된다. 메시지 형태는 이벤트 메시지를 나타내 준다. 개발자 고유 id는 클라이언트 호스트에서 서버로 연결된 각 개발자의 고유 식별자로써 쓰이기 위한 정보이다. 프로젝트 이름은 하나의 개발 프로젝트 내에 여러 서브 프로젝트가 있을 때 해당 프로젝트를 나타내기 위한 정보이다. 작성된 요소의 종류는 생성하려는 사용자 인터페이스 요소 - 텍스트 영역, 텍스트 필드, 버튼, 리스트, 라벨 - 에 대한 정보이고, 요소 속성은 각 요소가 갖는 특성 - 크기, 색, 폰트 - 을 나타내는 정보이다.

3.2.5 데몬

클라이언트로부터의 메시지는 단순한 텍스트 통신 메시지와 설계 이벤트 메시지이다. 서버는 각 클라이언트로부터 받은 메시지를 이용해 응용 프로그램의 원본을 만든다. 원본의 복사본은 공동 개발자가 속한 클라이언트 호스트로 브로드케스팅 된다. 이때 클라이언트가 가지는 사용자 인터페이스 응용 프로그램은 복사본이다. 이러한 일련의 과정을 처리하기 위해 서버와 클라이언트 호스트에는 데몬이 존재한다. 서버의 메시지 입력 데몬은 클라이언트로의 입력 메시지를 받아 메시지 종류를 판단하여 일관성(concurrency) 처리기로 전달하는 기능을 수행한다. 서버와 클라이언트간의 메시지 전송은 브로드케스팅 방법을 사용하므로 여러 개발자간 공유된 설계 요소의 일관성 유지를 위한 방법이 필요하게 된다. 서버의 이벤트 컨커런시(concurrency)데몬은 이러한 역할을 수행한다. 일관성 처리기 에서는 각 개발자가 발생한 이벤트 메시지를 조사하여 동일한 요소에 대한 이벤트일 경우 우선 처리 권한 모듈을 통해 순서를 정하고 순서에 따른 복사본을 만든다. 처리를 통한 메시지는 출력 처리 데몬을 통해 각 개발자에게 브로드케스팅 되어진다. 클라이언트의 메시지 입력 데몬은 서버로부터의 입력 메시지를 받아 개발자의 개발자 개인 창에 보여준다. 출력 데몬은 각 개발자가 발생한 통신 메시지나 작업 이벤트 메시지를 서버로 보내는 기능을 한다.

3.3 공동 개발 모듈

개발 모듈은 본 논문에서 제안된 통합 개발 환경(IDE -Integrated Development Environment)[15]의 실질적인 작업 공간이다. 공동 개발 창에서 이루어지는 실제적인 프로그램 설계 행위에 대해 자바 클래스로 작성된 사용자 인터페이스 도구 셋을 구성하는 클래스를 제공한다. 도구 셋은 아이콘형태로 제공된다. 초기 개발자가 서버로부터의 접근제어를 통해 지역 호스트에 연결한 후에 지역 호스트의 통신 데몬은 개발자에게 공동 개발을 위한 개발 툴과 개발자 개인 창을 보여주게 된다. 다음 (그림 4)는 공동 개발 창과 통신 개인 창 제어 모듈에 의해 생성된 공동 개발 창과 통신 개인 창이다.

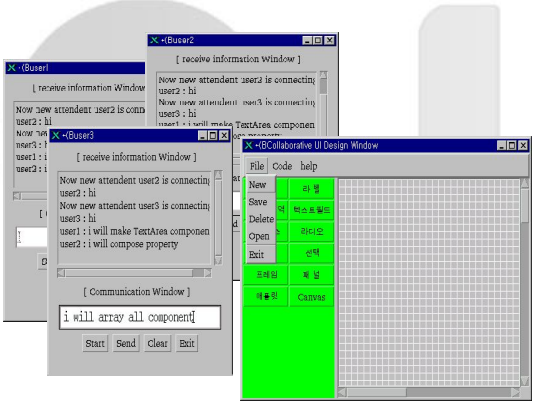

그림 4 공동 개발 창 과 통신 개인 창
Fig. 4 Co-development window and Individual window

다음은 개발 모듈을 이루고 있는 각 도구 셋 클래스중에 버튼을 생성하고 관련 실행모듈을 제작하기 위한 클래스 이다.

```
public class ButtonBuilder extends
  ComponentBuilder {
   public ButtonBuilder(int x, int y, int w, int h,
                        String name);
   public ButtonBuilder(int x, int y, int w, int h);
   public ButtonBuilder(int x, int y);
   public ButtonBuilder();
 }

class ComponentBuilder implement Drawable {
    public ComponentBuilder (int x, int y, int w,
                             int h, String name);
    public ComponentBuilder ();
 }
 interface Drawable();
 public class PositionLayout implements
 LayoutManger();
```

버튼 클래스는 다양한 형태의 버튼 인스턴스를 생성할 수 있는 여러 메소드 형태를 포함한다. 각 도구 클래스는 공동작업 창에 의해서 호출된다. 또한 생성된 요소에 대한 각 개발자의 접근은 작업제어 클래스에 의해서 제어된다. 호출된 도구 클래스는 공동 작업 창에 생성되고, 위치 관리자(LayoutManager)에 의해서 그려진다.

3.4 코드생성 모듈

시각 응용프로그램의 사용자 인터페이스 코드는 많은 부분에서 형태와 내용이 유사하기 때문에 한번 작성된 코드를 유사한 사용자 인터페이스의 재작을 위해 재사용하는 것은 전체 프로그램 개발 속도와 효율성을 향상 시킬 수 있다[16]. 코드생성 모듈은 개발자들이 아이콘(icon)형태의 개발 도구 셋을 이용하여 공동 개발한 사용자 인터페이스 설계 요소에 대해 자바(java)코드를 생성한다. 클라이언트에서 발생된 이벤트 메시지는 서버의 코드 생성기를 통해 각각 고유 클래스 이름이 붙은 간단한 형태의 자바 코드로 변환된다. 각 개발 도구 셋 클래스에서 생성된 요소에 대한 스트링 정보를 하나의 프로젝트 정보로 만든다. 이러한 정보는 설계된 사용자 인터페이스를 구성하는 모든 요소의 클래스 코드를 위한 정보이므로, 독립적인 사용이 가능하며, 다른 응용프로그램의 개발에도 사용될 수 있어서 코드의 재사용성을 높인다.

Ⅳ. 시스템 구현 및 실험

실험은 새로운 사용자 인터페이스를 공동 개발한다. 공동 개발 될 사용자 인터페이스 모델은 간단한 형태의 인터페이스 요소를 포함하는 GUI 화면이고, 독립적인 프로그램 모듈로서의 GUI 화면과 해당하는 java 코드가 결과로서 출력된다.

4.1 실험 환경

프로그램 공동 개발 환경의 구현은 Sun ultra2에서 SunOS5.5를 이용하였으며, JDK 1.3버전 환경에서 자바(java) 언어로 구현되었다. 각 개발자의 공동 작업을 위해 Sun ultra2와 Sun workstation server1000E를 이용하여 실험하였다.

4.2 공동 개발

실험을 통해 생성하려는 응용 프로그램의 결과 설계 화면은 (그림 5) 와 같다.

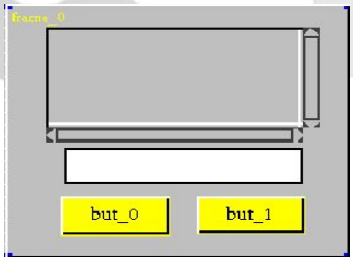

그림 5 사용자 인터페이스
Fig. 5 User Interface

프로그램 개발은 2명의 개발자가 참여한다. 개발 과정중에 발생하는 메시지는 (그림 6) 과 같은 통신 개인 창을 통해 개발자들에게 전달한다.

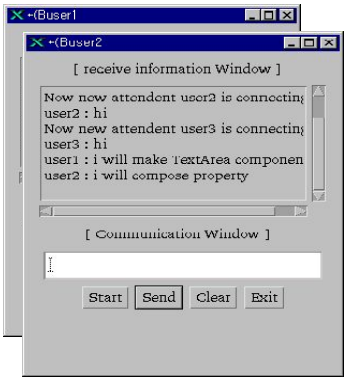

그림 6 공동 작업을 위한 개발자 통신 창
Fig. 6 Developer communication window for co-development

한 개발자는 생성될 화면을 구성하는 인터페이스 요소 생성, 속성 값 설정과 생성된 인터페이스 요소의 위치를 담당한다. 또 다른 개발자는 (그림 7) 과 같이 생성된 요소의 메소드를 선언하고 정의한다.

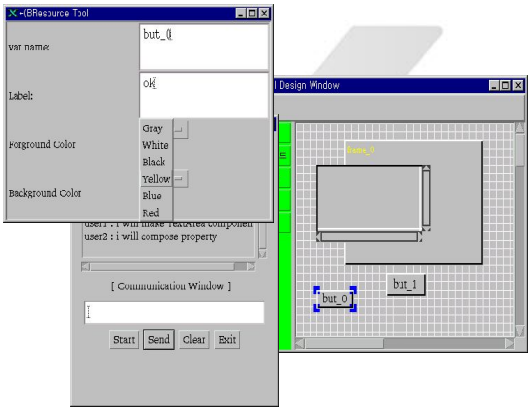

그림 7 화면 인터페이스 생성과 속성 부여
Fig. 7 Display interface generation and give a property

(그림 8) 과 같이 한 명의 개발자가 사용자 인터페이스를 설계하고 화면을 배치하면, 다른 한 개발자가 설계와 배치가 완료된 사용자 인터페이스의 각 구성 요소에 대해 자바 소스 코드 수준의 메소드를 입력한다. 따라서 완성된 응용 프로그램의 인터페이스와 메소드 부분에 대해, 소스 코드가 자바 코드로 출력된다. 생성된 자바 소스 코드는 다른 요소를 설계하고 개발할 때, 코드의 재사용 면에서 활용도가 높아 개발 시간과 노력을 줄여 프로그램 개발 생산성을 높일 수 있게 된다.

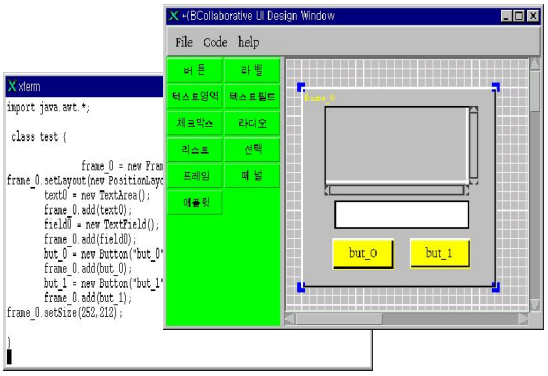

그림 8 생성된 인터페이스 요소와 자바 코드
Fig. 8 Generated interface attribute and Java code

V. 결론 및 향후 연구 방향

기존의 프로그램 개발 환경은 단일의 개발자가 모든 설계의 권한과 책임을 가진다. 따라서 규모가 크고 복잡한 프로그램의 개발에 있어 복잡한 인터페이스의 설계와 구현은 많은 부담이 된다. 여러 개발자의 공동 개발에 있어 기존의 방법은 작업의 독립성은 보장하지만, 협의를 통한 공동의 작업 공간을 이용해 즉각적인 화면 설계가 어렵다. 본 논문은 여러 개발자가 프로그램을 공동으로 개발할 때, 공동 작업 공간에서 협의를 통해 프로그램 개발이 이루어질 수 있는 공동 개발 환경을 설계하고 구현했다. 따라서 복잡한 프로그램의 개발에 있어 개발을 분담하고 네트워크를 통해 즉각적으로 개발자간 협의가 가능하여 보다 작업의 효율과 정확성을 높일 수 있다. 앞으로 제안된 시스템을 웹 기반으로 개발하여 보다 공간적, 시간적으로 활용도가 높고 제약성이 없는 공동 개발 환경으로 개발해야 할 것이다.

참고문헌

[1] Chales Rich and Candace Sidner, "Adding a Collaborative Agent to Graphical User Interfaces", ACM Symposium on User Interface Software Technology, 1996

[2] Brad A. Myers, "State of the Art in User Interface Software Tools", Advances in Human -Computer Interaction, Volume 4, 1993

[3] Brad A. Myers and Mary Beth Rosson, "Survey on User Interface Programming. Huma Factors in Computing Systems", Proceedings SIGCHI'92, 1992, pp. 192-202.

[4] Saul Greenberg, et al, "Issues an Experiences Designing and Implementing Two Gro Drawing Tools", Proceedings of Hawaii International Conference on System Sciences, pp. 138-150, 1993.

[5] Mark D. Gross, "Graphical Constraint sin CoDraw", Proceedings visual ranguages, 1992, pp. 81-87.

[6] Philip Isenhour, Clifford A. Shaffer, James Begole, Jeff Nielson, and Marc Abrams, "A Java-based Framework for Collaborative Interactive Modular Visualization Environments", Virginia University, TR-97-17, October 24, 1997.

[7] Guinn, C. I, "Mechanisms for Mixed-Initiative Human-Computer Collaborative Discourse", Proceedings 34th Annual Meeting of the Acl, 1996, pp.278-285.

[8] James Begole, Craig A. Struble, and Clifford A. Shaffer, "Collaboration Transparency in Java through Event Broadcasting," IEEE Internet Computing, Vol. 1, No. 2, March - April 1997.

[9] Henry Lieberman, "Autonomous Interface Agents", Proceedings SIGCHI'95, 1995, pp. 67-74.

[10] Ben Shneiderman, Designing the user interface, Addison Wesley

[11] H. M. Deitel and P. J. Deitel, Java how to program, Prentice hall

[12] Gary Cornell and Cay S. Horstmann, Core Java, The Sunsoft press

[13] Scott Oaks and Henry Wong, Java Threads, O'Reilly.

[14] David Flanagan, Java in a nutshell, O'Reilly.

[15] 박영만, 안영순, "복합 환경 통합 시스템 개발에 관한 연구", 한국컴퓨터정보학회 논문지, 제 5권, 제 1호, 2003

[16] 김홍진, "재사용을 위한 소프트웨어 부품구조에 관한 연구", 한국컴퓨터정보학회 논문지, 제5권, 제1호, 2000.

저 자 소 개

조 용 윤

1995년 시립 인천대학교 전자계산과 졸업(학사)

1998년 숭실대학교 대학원 전자계산학과 졸업 (공학석사)

1999~현재 숭실대학교 박사과정

<관심 분야> 컴파일러, 프로그래밍 언어, 프로그래밍 환경, XML

유 재 우

1976년 숭실대학교 전자계산학과 (공학사)

1978년~1987년 한국과학기술원 전산 학과(공학석사,공학사)

1986년~1987년 Cornell Univ. VisitingScientist

1983년~현재 숭실대학교 컴퓨터학과 정교수

<관심 분야> 컴파일러, 프로그래밍 환경, HCI, SGML, XML