IDE 기반의 분산 페어 프로그래밍 시스템의 설계 및 구현

박 지 훈[†] · 이 경 환[††]

요 약

물리적으로 분산된 개발자들은 분산 페어 프로그래밍 지원 시스템을 활용하여 페어 프로그래밍을 수행할 수 있다. 하지만 원격 화면 공유 기능 위주의 기존 CSCW 기반의 분산 페어 프로그래밍 지원 시스템들은 많은 개발자들이 분산 페어 프로그래밍을 수용하기에는 사용의 접근성, 용이성 등의 측면에서 미흡한 점이 많다. 본 논문에서 우리는 소프트웨어 개발자들이 분산 페어 프로그래밍을 수용하기 적합한 형태의 시스템 모델로서 IDE(Integrated Development Environment) 기반의 분산 페어 프로그래밍 시스템을 제안한다. 본 시스템의 GUI와 사용자 시나리오는 분산 페어 프로그래밍이 용이하도록 개발되었으며 디자인 패턴 을 적용하여 확장성이 높도록 시스템을 설계하고 자바언어로 구현하였다. 본 연구결과는 기존의 상업용 IDE에서 분산 페어 프로그래밍 기능과 GUI를 구현하는 개발자들에게 도움이 될 것이다.

Design and Implementation of Distributed Pair Programming System based on IDE

Ji Hoon Park[†] · Kyung Hwan Lee[††]

ABSTRACT

In distributed office, the pairs can program together using a distributed pair programming system. Many CSCW tools featuring remote screen sharing function have insufficient usability, accessibility to introduce many developers to distributed pair programming. In this paper, we suggest a distributed pair programming system based on IDE, which many developers will accept and use easily. We have developed a user scenario and GUI of the system, making distributed pair programming easier and designed with high extensibility by adapting design patterns and implemented in Java language. Our findings will be of significant help to developers dealing with implementation of distributed pair programming function into some commercial IDE.

키워드 : 페어 프로그래밍(Pair Programming), 분산 페어 프로그래밍(Distributed Pair Programming), 협업 소프트웨어 공학(Collaborative Software Engineering), 원격 교육(Distance Education), XP(eXtreme Programming)

1. 서 론

현대의 소프트웨어 개발팀들은 네트워 시대에 적합하게 지역적으로 분산되는 경우가 많아지고 있다. 그들은 인터넷과 통신망의 발달에 힘입어 원격으로 협업, 교육, 프로젝트 진행 등을 수행한다. 하지만 같은 장소에서 얼굴을 맞대고 작업을 진행하는 것보다는 개발 효율이 낮아서 프로젝트가 지연되거나 실패하는 경우도 많다. 이것을 막기 위해서는 효과적인 소프트웨어 협업 개발 체계를 구성해야 하며 현재 분산 페어 프로그래밍이 중요한 대안으로 부상하고 있다.

페어 프로그래밍(Pair Programming)이란 XP(eXtreme Programming)란 방법론을 통해 최근 널리 소개된 프로그래밍 방식으로서 한 컴퓨터에서 두 명의 프로그래머가 설계, 알고리즘, 코드, 테스트 등을 협력하여 완성하는 것이다[1]. 잘 수행된 페어 프로그래밍은 개발자의 생산성을 높이는 효과가 있다[12-14]. 분산 페어 프로그래밍(Distributed Pair Programming)이란 네트웍으로 연결된, 지역적으로 흩어진 소프트웨어 개발자들이 소프트웨어 프로그램의 도움을 받아 진행하는 페어 프로그래밍을 의미한다. 효과적인 분산 페어 프로그래밍은 지역의 차이로 발생하는 개발 효율성의 공백을 상당 부분 메꿀 수 있으며 이것은 지원 시스템이 제공하는 기능과 성능에 따라 결정된다[15].

현재 원격 화면 공유기능을 지원하는 일반적인 CSCW (Computer Supported Cooperative Work) 시스템이나 간단한 프로토타이핑 구현 수준의 분산 페어 프로그래밍 지원 시스템들이 있다. 그러나 그들은 원래 분산 페어 프로그래밍을 위한 용도로서 개발된 것이 아니기 때문에 사용의 접근성, 용이성 등의 측면에서 미흡하다. 따라서 지역적으로

† 준 회 원 : 중앙대학교 대학원 컴퓨터공학과
†† 정 회 원 : 중앙대학교 컴퓨터공학과 교수
논문접수 : 2004년 6월 11일, 심사완료 : 2004년 9월 6일

떨어져 있는 소프트웨어 개발팀들이 분산 페어 프로그래밍을 적극적으로 채택하지 못하는 것이 현실이다.

본 논문에서 우리는 소프트웨어 개발자들이 분산 페어 프로그래밍을 수용하기 적합한 형태의 시스템 모델로서 IDE(Integrated Development Environment) 기반의 분산 페어 프로그래밍 시스템을 제안한다. 본 시스템의 GUI와 사용자 시나리오는 분산 페어 프로그래밍이 용이하도록 개발되었으며 디자인 패턴 개념을 적용하여 확장성이 높도록 시스템을 설계하였다. 본 모델을 객관적으로 평가하기 위해 간단하지만 분산 페어 프로그래밍을 실제로 지원하는Coveloper For Java(이하 CFJ)란 시스템을 제시한다. 본 연구결과는 기존의 상업용 IDE에서 분산 페어 프로그래밍 기능과 GUI를 구현할 때 응용될 수 있다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장은 분산 페어 프로그래밍 지원 시스템을 이해하기 위해 필요한 기반연구들을 기술하고 3장은 우리가 제안한 지원 시스템의 소프트웨어 아키텍처, 4장은 그것을 구현한 CFJ의 실행 사례를 소개한다. 5장은 CFJ와 기존 분산 페어 프로그래밍 지원도구와의 비교평가, 마지막으로 6장에서 본 논문의 결론을 맺는다.

2. 관련 연구

2.1 페어 프로그래밍

페어 프로그래밍은 한 컴퓨터에서 두 명의 프로그래머가 설계, 알고리즘, 코드, 테스트 등을 협력하여 완성하는 프로그래밍 방식이다. 두 명중 한명인 드라이버(Driver)는 컴퓨터 앞에서 타이핑을 하거나 알고리즘을 적는다. 나머지 한 명인 파트너(Partner)는 그것을 관찰하며 오류를 발견하거나 대안을 생각한다[1]. 둘은 대화를 나누며 합의된 내용들을 구현한다. 중간에 드라이버와 파트너는 자주 그 역할을 교체하기도 한다.

이런 프로그래밍 방식이 대중적으로 널리 알려진 것은 최근에 주목 받고 있는 XP(eXtreme Programming)의 핵심 실천 방안(Core Practice)들 중의 하나이기 때문이다[1]. 페어 프로그래밍은 XP를 실천하면서그 효과가 극대화된다. XP의 다른 핵심 실천 방안들이 페어 프로그래밍과 밀접하게 연결되어 있기 때문이다.

페어 프로그래밍을 했을 때 얻을 수 있는 장점들은 페어 압력(Pair Pressure), 협의(Pair Negotiation), 용기(Pair Courage), 리뷰(Pair Reviews), 디버깅(Pair Debugging), 배움(Pair Learning), 믿음(Pair Trust) 등이 있다[2-5, 11].

2.2 분산 페어 프로그래밍과 지원 시스템

로컬 네트웍이나 인터넷을 이용하여 원격으로 페어 프로그래밍 하는 것을 분산 페어 프로그래밍이라고 한다. 페어 프로그래밍의 파트너가 바로 옆의 자리에 있지 않고 네트웍 너머에 있는 것이다. 이때 두 개발자는 서로 같은 화면을 공유하여 소스 편집, 컴파일, 실행의 모든 과정을 함께 할 수 있어야 한다. 한 개발자가 소스 코드를 키보드로 타이핑할 때마다 그 내용이 네트웍으로 떨어진 다른 사람의 컴퓨터로 즉시 전송되어 상대방의 화면에 그대로 보여야 한다. 다른 개발자는 소스 코드에서 에러가 발생하거나 개선할 부분이 있으면 채팅을 통해 상대방에게 알릴 수 있거나 또는 직접 소스코드를 수정할 수도 있다. 즉, 네트웍으로 연결된 두 개발자가 동시에 같은 소스 코드를 실시간 공유하여 공동으로 코딩하며 실행할 수 있다. 이때 두 사람은 서로 채팅이나 음성 통신 프로그램을 사용하여 대화를 나눈다.

최근 연구결과에 따르면 분산 페어 프로그래밍 역시 페어 프로그래밍의 효과를 근접하게 낼 수 있다[6].

2.2.1 분산 페어 프로그래밍과 CSCW 시스템

1960년대 엥겔바트(Engelbart)가 만든 NLS 시스템을 시작으로 CSCW 시스템은 현재까지 폭 넓은 연구가 진행되고 있다. 현재 효율적인 통신을 지원하는 하드웨어, 소프트웨어의 원격 공유를 위한 소프트웨어 아키텍처, 사람들간의 효율적인 커뮤니케이션 모델 등이 주로 연구되고 있다. 이들 중 소프트웨어의 원격 공유와 관련된 CSCW 시스템이 주로 분산 페어 프로그래밍에 자주 사용된다. 원격 공유의 소프트웨어 기능들은 화이트 보드, 화면 및 문서 공유, 멀티미디어 실시간 스트리밍(Real-time Streaming) 등이 있다. 이들 중 대중화된 것은VNC, PCAnywhere 등의 화면 전체를 원격으로 공유하게 해주는 소프트웨어이다. 전체 화면의 원격 공유는 관련 원격 공유 기술 중 상대적으로 간단하게 구현할 수 있기 때문이다. 따라서 이들 소프트웨어를 이용하여 많은 개발자들이 분산 페어 프로그래밍을 수행한다.

하지만 VNC(Virtual Network Computing), PCAnywhere 등의 CSCW 소프트웨어들은 처음부터 분산 페어 프로그래밍 지원을 위해 개발되지는 않았다. 다른 용도로 개발되었지만 원격으로 자원을 공유하고 음성 통신을 지원하는 기능 때문에 분산 페어 프로그래밍 도구로 사용되는 것이다.

분산 페어 프로그래밍에 자주 사용되는 대표적인 CSCW 관련 도구들의 분산 페어 프로그래밍과 관련된 장단점을 살펴보면 다음과 같다.

① VNC[7]

AT&T사가 만든 원격 화면 공유 소프트웨어이다. 인터넷에 연결된 어떤 컴퓨터 및 기기에서도 화면을 공유하고 통신할 수 있도록 개발되었다. 플랫폼 독립적이므로 서로 다른 운영체제에서도 화면을 공유할 수 있다. 주로 시스템 관리, IT 지원, 헬프 데스크의 용도로 사용된다.

전체 화면이 공유되어 한 사용자의 모든 화면의 움직임이 실시간으로 상대방에게 전송되어 보여지게 된다. VNC를 이

용한 분산 페어 프로그래밍 방식은 드라이버가 자신의 컴퓨터에서 IDE를 실행시킨 후 VNC를 통해 전체 화면을 상대방에게 전송하여 프로그램 개발 과정을 공유하는 것이다. 사용법이 간단하고 여러 운영체제에서 사용될 수 있는 장점이 있지만 페어 프로그래밍 드라이버의 프로그래밍과 관련 없는 화면 조작들이 걸러지지 않고 불필요하게 네트워크로 전송되는 단점이 있다. 또한 화면 이미지를 캡처하여 전송하기 때문에 송수신되는 데이터의 양이 문자 타입의 송수신보다 상대적으로 많다. 음성 데이터는 전송하지 못한다.

② NetMeeting

마이크로소프트사가 개발한 윈도우 계열의 실시간 협업 및 컨퍼런싱 소프트웨어이다. 전체 화면을 원격으로 공유할 수 있으며 텍스트 및 음성 채팅이 가능하다. 특히 특정 프로그램 화면만 공유할 수 있다. Netmeeting 을 사용한 분산 페어 프로그래밍은 전체 화면 또는 IDE 화면만 공유하여 음성 채팅을 하며 진행하는 방식이다. 하지만 원격 공유 속도가 다소 느리고 해상도의 제한이 있다. 또한 윈도우 계열의 운영체제만 서로 원격 공유가 가능하므로 분산 페어 프로그래머들은 동일한 MS 운영체제를 사용해야 하는 단점이 있다.

③ Speak Freely[8]

오픈 소프트웨어이며 무료인 인터넷 음성 채팅 소프트웨어로서 윈도우, 유닉스 운영체제에서 동작한다. 음성 채팅을 위한 전용 도구로서 분산 페어 프로그래밍에서는 주로 VNC의 음성 채팅 보조도구로서 사용된다. 음성 채팅은 텍스트 채팅보다 훨씬 효율적인 분산 페어프로그래밍 커뮤니케이션을 가능하게 한다.

④ QuantumPairs[10]

BigAtticHouse에서 개발한 윈도우 운영체제 계열의 분산 페어 프로그래밍 용 소스코드 에디터이다. 프로그램을 실행하면 소스코드 에디터가 화면에 뜨며 거기에 입력하는 모든 문자가 상대방에게 실시간으로 전달된다. VNC, Netmeeting 과 달리 화면을 캡처해서 상대방에게 전송하는 것이 아니라 입력된 문자를 그대로 실시간 전송하므로 네트워을 효율적으로 사용한다. 하지만 소스코드만 공유하며 코드의 컴파일, 실행 과정을 원격 공유할 수는 없다.

3. IDE 기반 분산 페어 프로그래밍 지원 시스템

3.1 시스템 아키텍처

대다수 개발자들이 소프트웨어를 개발할 때 IDE들을 사용하므로 그것에 기반한 분산 페어 프로그래밍 시스템은 기존의 분산 페어 프로그래밍 지원 시스템보다 사용의 용이성 면에서 높다.

본 논문에서 우리는 인터넷 환경에서 IDE(Integrated Development Environment) 기반 분산 페어 프로그래밍 지원 시스템의 확장성 높은 소프트웨어 아키텍처 모델을 제시한다. 그리고 그 모델을 구현한 시스템의 대표 이름을 Coveloper System이라 하고 본 연구의 결과로서 자바언어로 개발한 시스템을 CFJ로 명명한다.

인터넷 상의 소프트웨어 개발자들이 분산 페어 프로그래밍 기능을 수용한 IDE를 쉽게 수용하려면 다음과 같은 요구사항을 만족해야 한다.

ⓐ IDE에 익숙한 일반 개발자들이 이질감 없이 사용할 수 있는 분산 페어 프로그래밍 용 사용자 인터페이스와 프로그래밍 방식을 제공해야 한다.
ⓑ 분산 페어 프로그래머들이 효과적으로 커뮤니케이션할 수 있는 수단이 필요하다.
ⓒ 열악한 인터넷 통신망으로도 분산 페어 프로그래밍이 가능하도록 네트웍 송수신 데이터의 양이 작아야 한다.
ⓓ 분산 페어 프로그래밍으로 공유하는 데이터들은 실시간으로 빠르게 상대방에게 전달되어야 한다.

이런 요구사항들을 만족할 수 있도록 본 시스템은 다음과 같은 품질 목표를 달성해야 하며 소프트웨어 아키텍처는 본 목표를 만족하도록 개발되어야 한다.

ⓐ 높은 사용성(Usability)

분산 페어 프로그래밍 지원 기능은 IDE의 기존 GUI와 자연스럽게 통합되어야 한다.

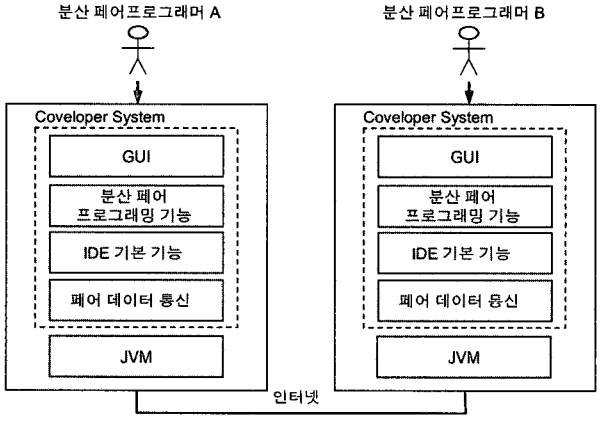
ⓑ 높은 확장성(Extensibility)

IDE에 새로운 기능이 추가되었을 때 그 기능도 쉽게 분산 페어 프로그래밍으로 공유 가능하도록 설계되어야 한다.

ⓒ 안정적인 실시간 서비스(Realtime)

분산 페어 프로그래밍 데이터가 손실되지 않고 실시간으로 송수신되어야 한다.

본 논문에서 개발한 CFJ는 레이어별로 (그림 1)과 같은 구조로 이루어져 있다.



(그림 1) Coveloper System의 레이어별 시스템 구조도

각 레이어별 세부 기능은 <표 1>과 같다.

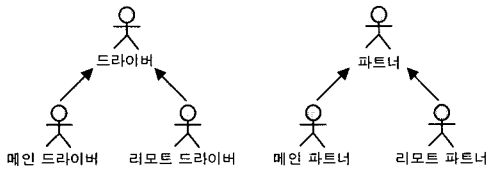
〈표 1〉 Coveloper System의 레이어별 세부 기능

시스템 레이어 이름	세부 기능
GUI 레이어	IDE의 기본 기능과 분산 페어 프로그래밍을 위한 사용자 인터페이스
분산 페어 프로그래밍 기능 레이어	분산 페어 프로그래밍 제어, 분산 소스코드 편집, 분산 컴파일, 분산 실행 기능
IDE 기본 기능 레이어	소스코드 편집, 컴파일, 실행 등의 일반적인 IDE 기능
페어 데이터 통신 레이어	분산 페어 프로그래밍 시에 데이터들을 인터넷으로 안정적이며 실시간으로 송수신하는 기능

3.2 CFJ 요구사항

3.2.1 개요

본 시스템을 사용하는 프로그래머는 (그림 2)와 같이 크게 드라이버와 파트너로 분류되고 내부적으로는 총 4개의 액터(Actor)로 역할이 나뉘어진다. 메인 리모트 드라이버



(그림 2) 액터의 종류

드라이버는 페어 프로그래밍에서 키보드를 점유하고 개발을 진행중인 자이고 파트너는 드라이버의 행위를 관찰하고 조언을 하는 자이다. 본 시스템에서는 분산 페어 프로그래밍 중에 "드라이버"라는 권한이 마치 토큰처럼 두 개발자 사이로 오고 가며 소스코드 편집권한이 이동된다.

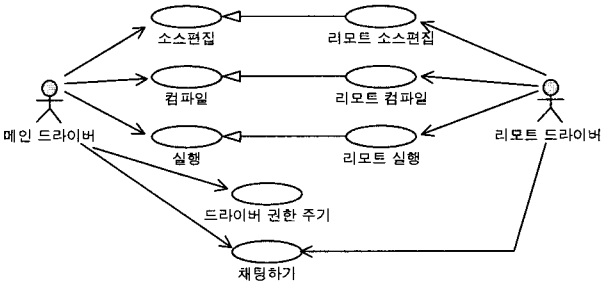
드라이버와 파트너는 분산 페어 프로그래밍의 초기 역할에 따라 각각 메인 드라이버, 리모트 드라이버 그리고 메인 파트너, 리모트 파트너로 세분화된다. "메인(Main)"은 분산 페어 프로그래밍 연결 작업을 할 때 먼저 요청한 개발자, "리모트(Remote)"는 요청을 받은 상대 개발자를 의미한다. "메인"과 "리모트"는 드라이버와 파트너의 역할을 서로 주고 받으면서 <표 2>와 같이 총 4개의 액터로 구분된다.

〈표 2〉 분산 페어 프로그래밍 개발자의 역할

액터 이름	역 할
메인 드라이버 (Main Driver)	분산 페어 프로그래밍을 처음 요청한 개발자가 "드라이버"역할을 수행할 때를 의미한다. 처음 상대방과 연결되어 분산 페어 프로그래밍 세션이 성립되었을 때 무조건 드라이버가 된다.
리모트 트너 (Remote Partner)	처음 상대방의 분산 페어 프로그래밍 요청을 승인한 개발자가 "파트너"역할을 수행하는 것을 의미한다.
메인 파트너 (Main Partner)	메인 드라이버가 리모트 파트너에게 "드라이버"역할을 넘겨줬을 때 메인 파트너는 "파트너"역할을 수행하는 메인 파트너가 된다.
리모트 드라이버 (Remote Driver)	리모트 파트너가 메인 드라이버로부터 "드라이버" 역할을 넘겨 받았을 때 "드라이버"역할을 수행하는 리모트 드라이버가 된다.

3.2.2 드라이버 유스케이스

메인 드라이버가 사용할 수 있는 기능을 유스케이스(use case)로 정리하면 (그림 3)과 같다.



(그림 3) 드라이버 관점의 유스케이스

각 유스케이스의 요약은 <표 3>과 같다.

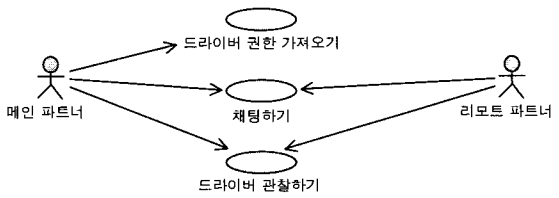
〈표 3〉 드라이버 관점의 유스케이스 요약

액 터	유스케이스 이름	요 약
메인 드라이버	소스편집	소스코드를 편집한다.
	컴파일	소스코드를 컴파일한다.
	실행	컴파일된 코드를 실행한다.
	드라이버 권한 주기	"드라이버"의 역할을 메인 파트너에게 준다.
	채팅하기	분산 페어 프로그래밍시에 리모트 파트너와 텍스트 채팅으로 대화를 나눈다.
리모트 드라이버	리모트 소스편집	상대방의 소스코드를 편집한다.
	리모트 컴파일	상대방의 소스코드를 컴파일한다.
	리모트 실행	상대방의 컴파일된 코드를 실행한다.
	채팅하기	분산 페어 프로그래밍시에 메인 파트너와 텍스트 채팅으로 대화를 나눈다.

소스코드의 보안 문제로 인해 메인 개발자만이 "드라이버" 역할 제어권을 가질 수 있다. 따라서 리모트 파트너가 "드라이버"의 권한을 갖기 위해서는 메인 드라이버에게 채팅으로 요청하여 메인 드라이버가 리모트 파트너에게 "드라이버" 권한을 줘야 한다.

3.2.3 파트너 유스케이스

파트너가 사용할 수 있는 기능을 유스케이스로 정리하면 (그림 4)와 같다.



(그림 4) 파트너 관점의 유스케이스

각 유스케이스의 요약은 <표 4>와 같다.

〈표 4〉 파트너 관점의 유스케이스 요약

액 터	유스케이스 이름	요 약
메인 파트너	드라이버 권한 가져오기	리모트 드라이버가 가지고 있는 '드라이버'역할을 가져온다. 그러면 메인 드라이버가 되고 상대방은 리모트 파트너가 된다.
	드라이버 역할 관찰하기	리모트 드라이버가 수행하는 소스코드 편집, 컴파일, 실행 등을 그대로 원격으로 관찰한다.
	채팅하기	분산 페어 프로그래밍시에 리모트 드라이버와 텍스트 채팅으로 대화를 나눈다.
리모트 파트너	드라이버 역할 관찰하기	메인 드라이버가 수행하는 소스코드 편집, 컴파일, 실행 등을 그대로 원격으로 관찰한다.
	채팅하기	분산 페어 프로그래밍시에 메인 드라이버와 텍스트 채팅으로 대화를 나눈다.

3.3 CFJ 설계

3.3.1 개요

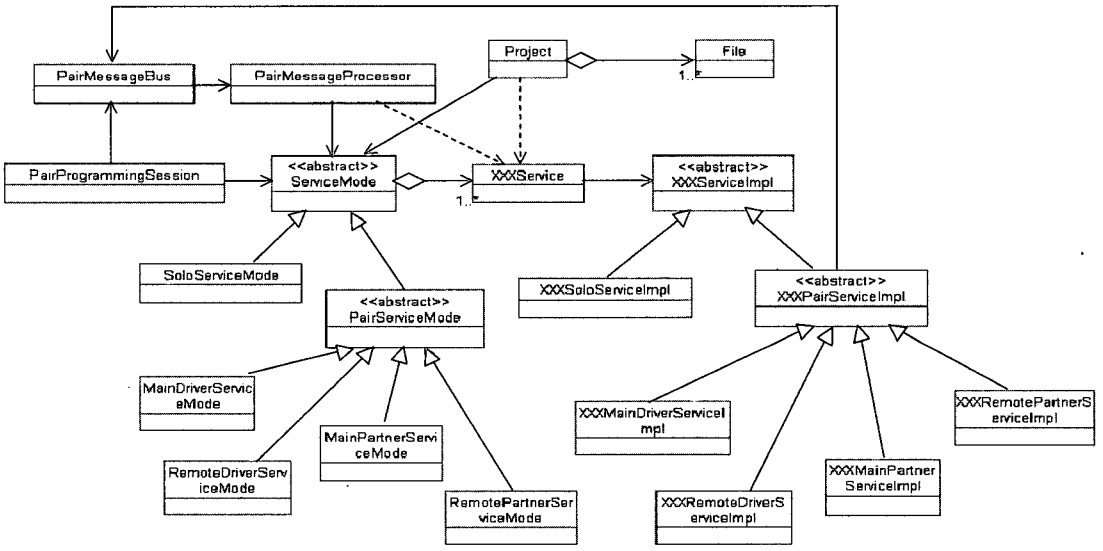
① 전체 클래스다이어그램

CFJ는 (그림 5)와 같이 자바언어로 구현된 20 여개의 핵심 클래스와 인터페이스들로 구성되어 있다.

각 핵심 클래스와 인터페이스들의 의미는 <표 5>와 같다.

〈표 5〉 핵심 클래스와 인터페이스의 의미

클래스 이름	역 할
PairProgrammingSession	인터넷을 통한 분산 페어 프로그래밍 세션(Session)을 의미한다. 분산 페어 프로그래밍 요청을 상대방이 승인하면 그 순간 생성되고 페어 연결이 종료되면 사라진다.
PairMessageBus	분산 페어 프로그래밍 세션이 성립된 후 페어 데이터들을 네트웍에 송수신하는 기능을 수행한다.
PairMessageProcessor	PairMessageBus로부터 받은 페어 데이터를 해석하여 적절한 Service 객체를 찾고 데이터를 전달하는 기능을 수행한다.
ServiceMode	IDE의 현재 상태가 솔로 프로그래밍, 분산 페어 프로그래밍인지에 따라 관련 Service들의 집합을 생성하는 인터페이스 역할을 한다. 하위 클래스인 SoloServiceMode, MainDriverServiceMode, MainPartnerServiceMode, RemoteDriverServiceMode, RemotePartnerServiceMode 등이 이 인터페이스를 구현한다.
XXXService	소스코드 편집, 컴파일, 실행 등의 IDE의 기본 서비스(Service)들을 상징한다. XXX는 컴파일, 실행 등의 서비스 이름을 의미한다. 실제로는 IDE 기본 기능의 인터페이스만 있고 XXXServiceImpl 인터페이스의 구현 객체가 해당 기능을 위임(delegation)받아 구현한다.
XXXServiceImpl	XXXService의 구현 객체의 인터페이스로서 이 클래스의 하위 클래스들이 실제로 해당 서비스를 구현한다.
Project	의미있는 파일들의 집합체이다.
File	공유가능한 소스코드나 텍스트 파일등을 의미한다.



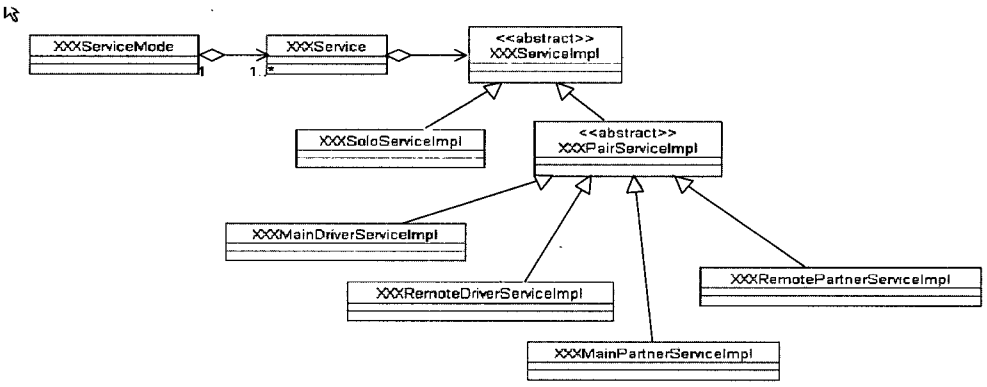
(그림 5) CFJ 의 클래스 다이어그램

② Service 클래스와 ServiceImpl 클래스의 관계

위 클래스 중에서 특히 Service 클래스와 ServiceImpl 클래스는 확장성에 있어서 중요한 역할을 수행한다. (그림 6)에서 이들은 Strategy 디자인 패턴이 적용되었다[9].

분산 페어 프로그래밍에 참여하는 개발자들은 시간에 따라 4개의 액터들, 즉 메인드라이버, 메인파트너, 리모트드라이버, 리모트파트너 들로 역할이 변경되고 그때마다 시스템이 제공하는 서비스(소스코드 편집, 컴파일, 실행 등)들의 행위가 달라지기 때문에 그것을 수용하려면 Strategy 패턴이 적절하다. 예를 들어 어떤 개발자가 메인 드라이버 역할일 때 컴파일을 하면 그 결과가 상대방에게 네트웍으로 전송되어야 한다. 반면 그 개발자가 "드라이버"역할을 상대방에게

주어서 메인 파트너가 되었다면 메인 파트너는 컴파일을 할 수 없어야 한다. 왜냐면 파트너는 드라이버의 관찰자이기 때문이다. 이렇듯 서비스들은 개발자의 역할이 변경될 때마다 행위가 달라지게 된다. 달라지는 서비스의 행위들은 XXXServiceImpl의 하위 클래스들이 구현하고 XXXServiceMode가 개발자의 역할에 따라 그들을 변경함으로서 분산 페어 프로그래밍의 가변적인 서비스 행위를 처리할 수 있게 되었다.



(그림 6) Service와 ServiceImpl 클래스 다이어그램

3.3.2 분산 컴파일 서비스

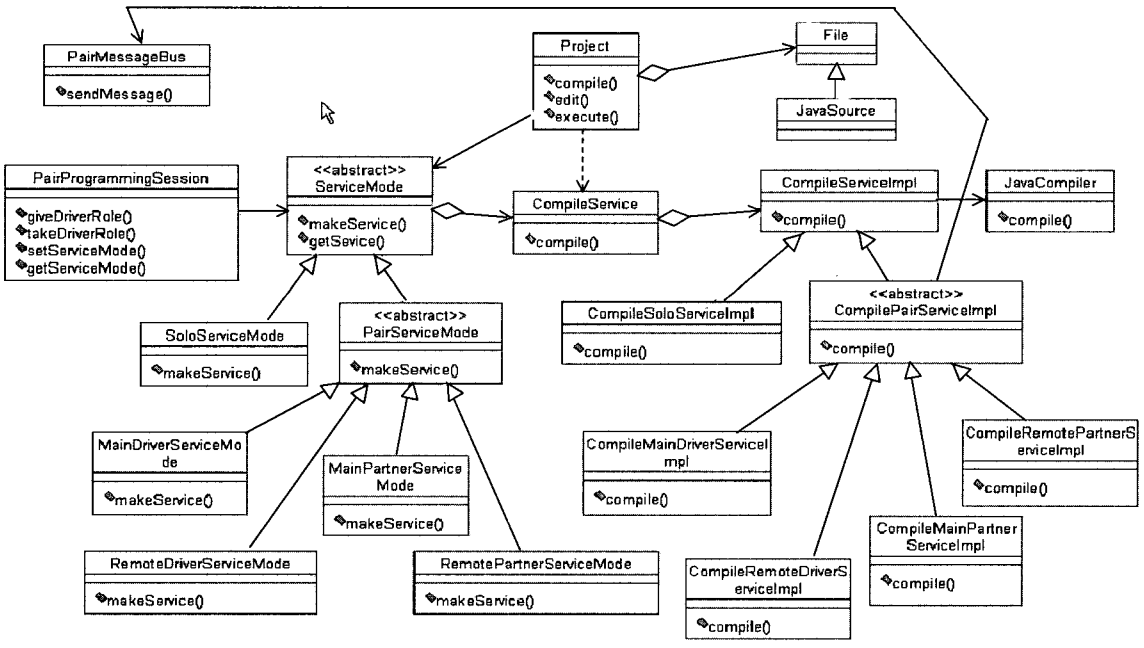
분산 페어 프로그래밍 세션이 성립되지 않은 상태, 즉 솔로 프로그래밍(Solo Programming)상태의 컴파일 서비스와 세션이 성립된 이후의 동일 서비스는 각 개발자의 역할에 따라 행위가 달라진다.

컴파일 관련 클래스들의 메소드를 정의한 클래스 다이어그램은 (그림 7)과 같다.

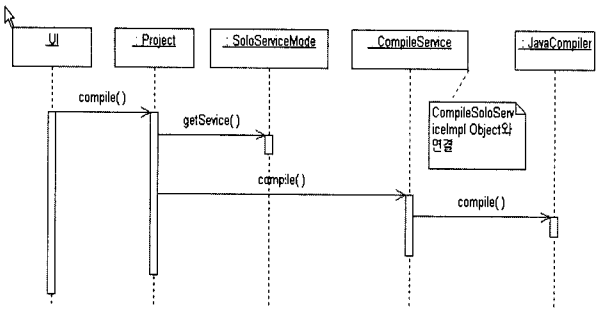
① 솔로 컴파일 시퀀스 다이어그램

먼저 분산 페어 프로그래밍 세션이 연결되지 않은 상태에서, 즉 로컬 IDE 상태의 컴파일 서비스는 (그림 8)과 같이 수행된다.

GUI의 컴파일 버튼을 누르면 현재 사용자가 선택한 Project 객체가 현재 서비스 모드를 반환한다. 현재는 분산 페어 프로그래밍 세션이 성립되지 않은 상태이므로 SoloServiceMode 객체는 로컬에서 컴파일하는 서비스의 구현 객체인 CompileSoloServiceImpl 객체를CompileService 객체에게 연결시켜 준다. CompileSolServiceImpl 객체는 로컬의 JavaCompiler에게 컴파일 요청을 하게 된다. 따라서 IDE는 로컬에서 컴파일한 결과를 로컬 화면에서만 그 결과를 보여주게 된다.



(그림 7) 컴파일 클래스다이어그램



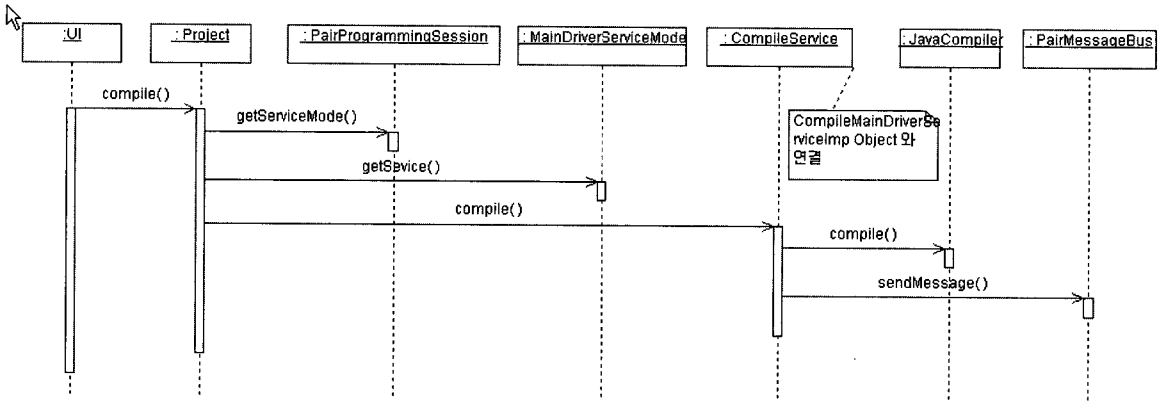
(그림 8) 솔로 컴파일 시퀀스다이어그램

② 페어 컴파일시퀀스 다이어그램

그렇지만 분산 페어 프로그래밍 세션이 연결된 이후에는 컴파일 서비스의 행위가 달라지게 된다. 메인 드라이버가 컴파일을 한다고 가정해보자. 이때는 컴파일 결과가 네트웍 상대방인 리모트 파트너에게 실시간으로 전달되어야 한다.

○ 메인 드라이버 관점

메인 드라이버가 컴파일 버튼을 눌렀을 때의 이벤트 처리 시퀀스 다이어그램은 (그림 9)와 같다.

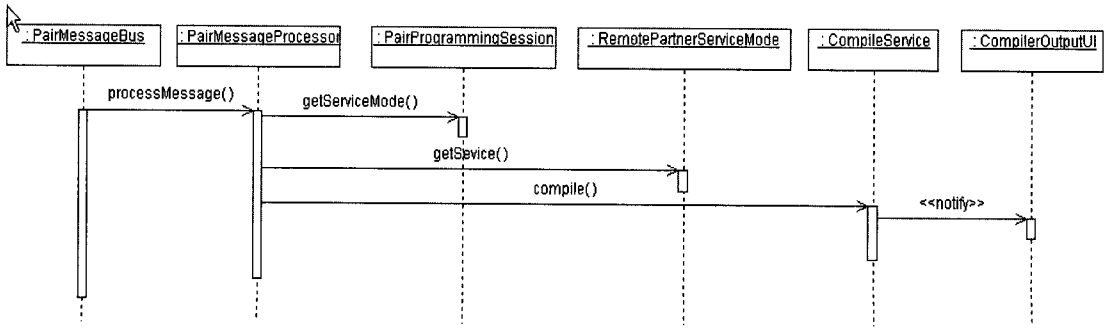


(그림 9) 메인드라이버의 컴파일 시퀀스 다이어그램

사용자가 현재 선택한 Project 객체는 분산 페어 프로그래밍 세션인 PairProgrammingSession 객체로부터 현재의 SerivceMode 객체를 반환 받는다. 이때 개발자는 메인 드라이버이기 때문에 MainDriverServiceMode 객체를 반환 받고 이것은 CompileService와 CompileMainDriverServiceImpl 객체를 연결시켜준다. CompileMainDriverServiceImpl 객체는 JavaCompiler에게 컴파일을 요청한 후 그 결과를 PairMessageBus를 통해 리모트 파트너에게 전송하는 역할을 한다. 따라서 메인 드라이버의 컴파일 서비스는 로컬에서 컴파일을 수행한 후 상대방에게 그 결과를 실시간으로 보내어 메인 파트너가 메인 드라이버의 컴파일 진행상황을 관찰하게 해준다.

• 리모트파트너 관점

리모트 파트너는 메인 드라이버가 컴파일한 결과를 PairMessageBus로부터 받는다. 이때의 진행과정은 (그림 10)과 같다. PairMessageBus는 해당 메시지를 PairMessageProcessor에게 넘긴다. 이 클래스는 컴파일 메시지를 해석하고 Compile 서비스를 실행한다. 그 전에 PairProgrammingSession으로부터 Remote Partner Service Mode 객체를 먼저 넘겨받는다. 참고로 ServiceMode의 종류는 "드라이버" 역할을 주고 받을 때 결정되고 유지된다. RemotePartnerService Mode 객체는 CompileService 객체와 메인 드라이버로부터 받은 컴파일 결과 메시지를 화면에 뿌려주는 CompileRemotePartnerServiceImpl 객체를 연결시켜 준다. 즉 컴파일 결과 창을 상징하는 CompilerOutputUI 객체에게 컴파일 결과 메시지를 출력하여 메인 드라이버가 컴파일한 결과를 리모트 파트너는 관찰할 수 있다.

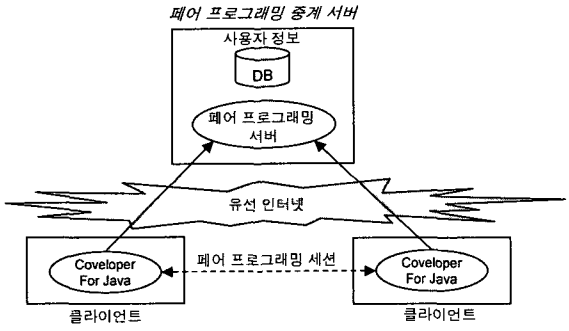


(그림 10) 리모트파트너의 컴파일 시퀀스 다이어그램

4. CFJ의 구현 및 실행

4.1 소 개

IDE 기반의 분산 페어 프로그래밍 지원 모델을 자바언어로 구현한 CFJ는 인터넷상의 어떤 누구와도 같은 소스코드를 공유하여 동시에 편집하고 컴파일하고 실행할 수 있도록 P2P 네트웍 기능이 포함된 실험용 IDE이다.



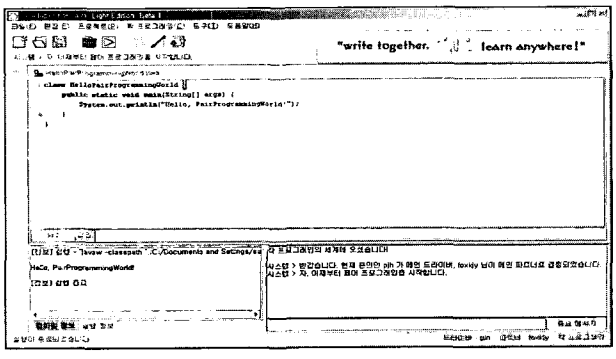
(그림 11) CFJ 시스템 구조도

CFJ는 클라이언트/서버 구조로 되어 있으며 분산 페어 프로그래밍의 상대방을 찾고 세션을 성립할 때 중개 서버를 이용하고 세션이 성립된 이후에는 상대방과 직접 네트웍 연결이 되어 데이터를 주고 받게 된다.

클라이언트와 서버의 실행환경은 MS사의 윈도우 98/ME/NT/2000, Linux, HP-UX 등의 자바가상머신(Java Virtual Machine)이 동작하는 컴퓨터이며 자바 1.4 버전 이상이어야 한다.

4.2 실행 예

분산 페어 프로그래밍이 성립되면 (그림 12)와 같이 인터넷을 통해 소스코드를 실시간으로 같이 공유하게 된다.



(그림 12) 분산 페어 프로그래밍이 연결된 이후의 모습

이때부터 상대방은 소스코드 편집기에서 입력하는 모든 문자들을 실시간으로 보게 된다. 또한 오른쪽 아래에 생긴 채팅창으로 대화를 나눌 수 있다. 컴파일, 실행을 하면 왼쪽 아래창에 보이는 결과가 상대방에게 그대로 전송이 되어 실시간으로 공유하게 된다. 마치 한 대의 컴퓨터에 두 명이 의자에 앉아 모니터를 주시하는 "페어 프로그래밍"과 유사한 방식을 인터넷상에서 구현한 것이다.

5. 시스템 평가

위의 "3.1 시스템 아키텍처"에서 제시한 분산 페어 프로그래밍 지원 도구의 품질 평가 기준인 분산 페어 프로그래밍의 용이성, 안정적 실시간 서비스, IDE 기반의 분산 페어 프로그래밍 기능의 확장성 및 플랫폼 독립성의 관점에서 CFJ와 기존의 CSCW 툴을 비교하였다.

5.1 분산 페어 프로그래밍의 용이성

CFJ는 IDE에 익숙한 일반 개발자들이 이질감 없이 분산 페어 프로그래밍을 할 수 있도록 IDE 내에 분산 페어 프로그래밍 용 GUI와 기능을 내포하였다. 따라서 개발자들은 언제든지 프로그램을 개발하는 중간에 IDE 내에서 분산 페어 프로그래밍을 즉시 사용할 수 있으므로 사용의 용이성이 높다. VNC, Netmeeting 등을 사용하여 분산 페어 프로그래밍을 하려면 IDE와 별도의 프로그램 실행 및 셋업 과정이 추가로 필요하다. QuantumPairs는 분산 컴파일, 실행이 되지 않고 소스코드 편집만 원격 공유가 가능하다.

5.2 안정적 실시간 서비스

분산 페어 프로그래밍으로 공유하는 데이터들은 실시간으로 빠르게 상대방에게 전달되어야 한다. CFJ는 상대방과 P2P 방식으로 통신을 하며 화면 이미지를 캡처하지 않고 텍스트와 위치 데이터만을 전송하므로 분산 소스코딩 공유와 같은 실시간 서비스가 무리 없이 가능하다. VNC, Netmeeting은 화면 이미지 캡처 데이터를 전송하므로 CFJ보다 네트웍 전송 속도 느리다. QuantumPairs는 CFJ와 동일한 네트웍 전송 방식이므로 네트웍 속도가 빠르다.

5.3 분산 페어 프로그래밍 기능의 확장성

CFJ는 새로운 분산 페어 프로그래밍 공유 서비스가 추가되어도 XXXServiceImpl을 구현한 새로운 클래스만 추가하면 기존 코드의 수정을 최소화할 수 있다. 즉 IDE에 새로운 기능이 추가되었을 때 그 기능도 쉽게 분산 페어 프로그래밍으로 공유 가능하도록 설계되어 있다. VNC, Netmeeting은 단순 원격 화면 공유기 이므로 비교 대상에서 제외하며, QuantumPairs는 소스코드 편집만 원격으로 공유되고 컴파일, 실행 등은 공유되지 못한 구조로 봤을 때 CFJ보다 확장성이 낮다.

5.4 플랫폼 독립성

CFJ는 자바로 구현되었고 텍스트와 위치 데이터만을 전송하므로 플랫폼 독립성이 높다. VNC는 윈도우, 유닉스 계열의 프로그램이 존재하며 Netmeeting, QuantumPairs는 윈도우 계열의 PC에서만 동작한다.

위에서 기술한, CFJ와 다른 분산 페어 프로그래밍 지원 도구의 비교 내용을 <표 6>에 정리하였다. (Speak Freely는 음성 채팅 전문 도구이므로 제외하였다)

〈표 6〉 CFJ와 기존 분산 페어 프로그래밍 지원 도구의 비교 평가

비교 항목	분산 페어 프로그래밍의 용이성	안정적실시간 서비스	IDE 기반 분산 페어 로그래밍 기능의 확장성	플랫폼 독립성
CFJ	상	상	상	상
VNC	중	중	N/A	상
Netmeeting	중	중	N/A	하
QuantumPairs	중	상	하	하

6. 결 론

UML로 IDE 기반의 분산 페어 프로그래밍 지원 시스템을 모델링하고 자바언어로 구현하였다. 확장성이 높고 사용성이 높은 시스템을 목표로 개발하였으며 또한 일부 개발자들을 대상으로 IDE 기반의 분산 페어 프로그래밍의 효과를 실험하였다.

연구결과로서 다음과 같은 결론을 얻었다.

첫 번째, 분산 페어 프로그래밍을 원하는 개발자는 기존의 VNC, PCAnywhere등과 같이 다른 용도의 분산 페어 프로그래밍 지원 도구보다 IDE 기반의 방식을 선호한다는 사실을 확인하였다. 개발자가 매일같이 접하는 자신의 IDE를 통해 자연스럽게 분산 페어 프로그래밍을 할 수 있다면 진입장벽이 낮춰질 것이다.

두 번째, IDE의 각종 기본 서비스(소스코드 편집, 컴파일, 실행 등)를 분산 페어 프로그래밍으로 손쉽게 공유할 수 있도록 객체 지향의 설계 원리를 적극적으로 활용하여 확장성이 뛰어나도록 설계하였다. 상용 IDE인 이클립스나 J빌더(JBuilder) 등에도 응용하여 사용할 수 있도록 높은 추상화 레벨을 적용하였다.

세 번째, 분산 페어 프로그래밍에 참여하는 사용자들은 "드라이버"의 권한에 따라 상태가 달라지고 시스템의 행위가 달라지는데 그것을 Strategy 디자인 패턴을 사용했을 때 유연하게 대처할 수 있었다.

네 번째, 개발자들끼리의 커뮤니케이션 수단으로는 텍스트 채팅보다 음성채팅이 더욱 효과적이다. Speak Freely와 같은 음성 채팅 기능이 추가되면 좀 더 효율적인 분산 페어 프로그래밍이 될 것이다.

다섯 번째로 분산 페어 프로그래밍 지원도구를 온라인 프로그래밍 교육에 활용해도 좋은 효과가 있다는 사실을 알았다. CFJ 를 사용하여 간단한 온라인 프로그래밍 교육 실험을 하였을 때 프로그래밍 스킬이 향상되는 효과가 있었음을 확인하였다.

향후 분산 페어 프로그래밍으로 공유하는 정보들의 보안과 각 참여자의 역할 변경에 따른 권한 설정 관련의 추가 연구가 필요하다.

참 고 문 헌

[1] Kent Beck, Extreme Programming Explained, Addison-Wesley, 2000.

[2] Williams, Laurie and Robert Kessler, Pair Programming Illuminated, Addison-Wesley, 2002.

[3] L. A. Williams, The Collaborative Software Process, PhD Dissertation., Department of Computer Science, University of Utah. Salt Lake City, 2000.

[4] Alistair Cockburn and Laurie Williams, "The costs and benefits of pair programming," In Giancarlo Succi and Michele Marchesi, editors, Extreme Programming Examined, pp.223-247. Addison-Wesley, 2001.

[5] Laurie A. Williams, "Strengthening the case for pair programming," IEEE Software, pp.19-25, July-August, 2000.

[6] Baheti, Prashant, Edward Gehringer and David Stotts, "Exploring the Efficacy of Distributed Pair Programming," Proceedings Extreme Programming and Agile Methods -XP/Agile Universe, 2002, pp.208-220, August, 2002.

[7] AT&T Laboratories Cambridge, VNC(Virtual Network Computing), http://www.realvnc.com/, current Jun., 2004.

[8] Brian C. Wiles, Speak Freely, http://www.speakfreely.org/, current, Jun., 2004.

[9] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, Design Patterns, Addision Wesley, 1995.

[10] BigAtticHouse, QuantumPairs, http://www.bigattichouse.com/, current, Jun., 2004.

[11] Hanks, Brian, Empirical Studies of Pair Programming, 2nd International Workshop on Empirical Evaluation of Agile Processes (EEAP 2003), August, 2003.

[12] Sven Heiberg, Uuno Puus, Priit Salumaa and Asko Seeba, Pair-Programming Effect on Developers Productivity, Proceedings of XP2003 (Springer LNCS 2675), pp.215-224, 2003.

[13] Jensen, Randall W., A Pair Programming Experience, CrossTalk, The Journal of Defense Software Engineering, March, 2003.

[14] Thomas, L., M. Ratcliffe and A. Robertson, Code Warriors and Code-a-Phobes : A Study in Attitude and Pair Programming, Proceedings of SIGCSE 2003, pp.363-367, February, 2003.

[15] Janes, Andrea, Barbara Russo and Giancarlo Succi, Use of Pair Programming for Experience Exchange in a Distributed Internship Project - A preliminary analysis of the results, OOPSLA2002 workshop on Pair Programming Explored, November, 2002.

박 지 훈

e-mail : pjh@object.cau.ac.kr
1995년 중앙대학교 전자계산학과(학사)
1997년 중앙대학교 컴퓨터공학과 대학원 (공학 석사)
1997년~현재 중앙대학교 컴퓨터공학과 박사과정
관심분야 : CBD, SPI, 소프트웨어 아키텍처, Agile 소프트웨어 개발

이 경 환

e-mail : kwlee@object.cau.ac.kr
1964년 중앙대학교 이과대학 수학과(학사)
1966년 중앙대학교 이과대학 수학과(석사)
1980년 중앙대학교 대학원 수학과(박사)
1982년~1983년 미국 AUBURN 대학교 객원교수
1986년~1986년 독일 BONN 대학교 객원교수
1992년~JCSE '92 Conference Chairman 서울
1993년~1995년 중앙대학교 공과대학 학장
1994년~1995년 중앙대학교 정보산업대학원 원장
1998년~2000년 한국 온라인 가상 대학 운영 위원장
1998년~2000년 중앙대학교 정보통신연구소 소장
1999년~2000년 한국 정보과학회 회장
2000년~2001년 중앙대학교 총무처장
2001년~2002년 미국 USC 대학교 객원교수
1992년~현재 ISO/SC7/WG10 SPICE 한국 위원장
1992년~현재 한국표준 소프트웨어 공학(SC7) 위원
관심분야 : CBD Architecture, SPI(Defect Analysis), MBASE/CeBASE