DOI: 10.3745/KIPSTD.2009.16-D.4.571

협업개발 환경에서의 효율적 동시성 제어를 위한 P2P기반 충돌해결 기법

박 현 수[†] · 김 대 엽[††] · 윤 청[†††]

요 약

본 논문은 구성원들 사이의 자원공유를 원천적으로 봉쇄하는 일반적 협업개발 지원도구의 한계를 극복하고, 조직의 여러 사용자들에게 자원 공유의 기회를 제공함과 동시에 자원공유에서 발생하는 버전관리와 충돌문제를 해결하기 위한 방법을 제시한다. 개발된 소프트웨어 협업개발 지원도구는 전통적인 낙관적 기법을 적용하되 충돌해결에 드는 비용과 노력의 절감을 위한 개선된 알고리즘을 사용한다. 시스템의 구조는 전통적인 Client/Server 방식에 개인간 정보교환을 지원하는 P2P(peer-to-peer) 방식이 결합된 형태로 이루어져 있으며, 공개 소프트웨어인 CVS(Concurrent Version System)를 기반으로 구현되었다. 이를 바탕으로 대표적인 기존 협업개발 지원도구들과의 유용성 비교를 통해 기능적 효율성을 확인하였다.

키워드: P2P, 협업개발, 동시성 제어, CVS, 소프트웨어 형상관리

P2P Based Collision Solving Technique for Effective Concurrency Control in a Collaborative Development Environment

Hyun Soo Park[†] · Dae Yeob Kim[††] · Cheong Youn[†††]

ABSTRACT

This paper provides a way to overcome limitations of general collaborative software development tools that completely restrict co-ownership of resources among individuals in a team oriented developmental environment. It also provides a solution for users to co-own resources and at the same time manage version control and collision problems that may occur due to the co-ownership of resources.The cooperative development support tool of developed software uses the conventional optimistic technique but employs the algorithm which is improved to reduce costs and efforts required for solving collision. The collaborative software development tool presented in this paper is made up of the classical client/server structure with the P2P(peer to peer) method which supports information exchange among individuals. This tool is developed based on open source software CVS(Concurrent Version System). Functional efficiency was confirmed by comparing it to the utility of prior existing collaborative software development tools.

Keywords : P2P, Collaborative Development, Concurrency Control, CVS, Software Configuration Management

1. 서 론

e-비즈니스 시대에 접어들어 소프트웨어 개발 생명주기가 점점 짧아지고 그 활용분야가 산업 전 분야로 확산되면서 소프트웨어 품질의 중요성은 더욱 높아지고 있다. 또한 소프트웨어에 대한 사용자들의 서비스 요구사항이 빠르게 변화하여 신속하고 정확하게 그 요구사항을 반영할 수 있는 개발조직의 역량이 필요하다. 이렇듯 소프트웨어 시장의 빠른 변화에 대응하기 위해 소프트웨어 개발조직은 점점 대형화 되어가고 있으며, 나아가 소프트웨어 개발환경의 분산화 또한 확산되고 있다. 이러한 환경의 변화에 따라 소프트웨어 개발과 관련된 거의 전 분야에 걸쳐 소프트웨어의 품질을 향상시키고 인터넷을 이용한 상호 정보교류와 의사소통을 지원하는 소프트웨어 협업개발 지원도구의 수요가 증가하고 있는 추세이다[1]. 또한 개인의 역량이 아닌 조직적 역량 차원에서 소프트웨어의 품질을 관리하려는 시도가 점차 늘어감에 따라 팀 단위 작업 시 요구되는 협업기능의 중요성이 점점 높아지고 있다.

팀 단위의 개발환경에서 소프트웨어 협업개발을 원활히 수행하기 위해서는 구성원 간의 정보 공유와 유기적인 의사

† 정 회 원 : 백석문화대학 컴퓨터정보학부 부교수
†† 준 회 원 : 충남대학교 컴퓨터공학과 석·박사 통합과정
††† 정 회 원 : 충남대학교 전기정보통신공학부 교수
논문접수 : 2009년 5월 4일
수 정 일 : 1차 2009년 5월 27일, 2차 2009년 6월 5일
심사완료 : 2009년 6월 5일

소통 기능이 요구된다. 특히 여러 개발자들이 참여하는 대규모 프로젝트의 경우 개발 생산성을 향상시키기 위해서는 동일 자원에 대한 병렬개발의 지원이 필요하다. 또한 분산된 환경에서 자원에 대한 접근성을 제공함으로써 시간과 공간의 제약을 극복할 수 있도록 해야 한다. 하지만 병렬 개발의 경우 특정 자원을 두 명 이상의 사용자가 동시에 접근 및 수정함으로 인해 충돌문제가 발생할 수 있다. 충돌이란 특정 자원 즉 파일이 동시에 여러 명의 사용자에 의해 수정되고 있음을 의미한다. 이는 파일 내의 특정 문맥에 대한 동시 수정을 의미하는 것은 아니며 서로 다른 문맥에 대해 수정하더라도 해당 파일에 대해서는 충돌이 발생했다고 할 수 있다. 충돌의 발생은 자원에 대한 일관성을 저해하는 요소로 작용할 수 있다. 각자의 작업 결과는 서버에 저장되어 있던 기존의 내용과 모두 다른 모습을 취하고 있을 수 있으므로 일관성의 유지를 위해서는 이러한 결과를 하나의 모습으로 통합하여 서버에 반영할 수 있는 기법이 반드시 필요하다. 또한 작업의 중복으로 인한 자원 충돌문제는 그것을 인식하는 시점이 늦어지는 경우 인적자원의 낭비를 초래할 수 있으므로 실시간으로 수행되는 협업개발 환경에서 구성원들의 병렬작업을 시간의 지연 없이 동기화할 수 있는 동시성 제어기법(concurrent access control)이 필수적으로 요구된다[2, 12-14].

본 논문은 효율적인 병렬작업을 지원하기 위해 자원의 소유권을 특정 사용자에게 국한시키지 않고 모든 사용자가 자유롭게 그 복사본을 수정할 수 있도록 허용하며, 동시 작업의 결과로 발생할 수 있는 충돌문제를 보다 이른 시간에 해결하여 자원의 일관성을 유지할 수 있도록 하는 기법을 제시한다. 복사본이란 서버에 저장되어 있는 원래의 자원을 자신의 로컬에 다운로드 한 것을 의미하며 이를 로컬 복사본(local copy)라 한다[12-14].

동시 작업과 충돌문제의 해결을 위해서 낙관적 동시성 제어기법을 변형한 P2P 기반의 충돌해결 기법을 제안하며 이는 다음과 같은 기능을 가지고 있다.

1) 공유 자원에 대한 동시 수정(충돌) 여부를 주기적으로 감시
2) 충돌이 감지된 경우 현재 자원을 공유하고 있는 사용자들에게 통보
3) 서버를 거치지 않고 사용자들 간에 곧바로 충돌을 해결할 수 있도록 지원

위의 기능들을 통해 보다 이른 시간에 동시 작업으로 인한 충돌을 인식하고 해결할 수 있도록 하여 자원의 일관성을 유지하는데 드는 비용을 절약하고자 하는 것이 본 연구의 목적이라 할 수 있다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에서는 관련 연구로 전통적인 동시성 제어기법과 연구 사례를 소개하고 일반적인 소프트웨어 협업개발 지원도구가 갖춰야 할 기능적 요소들에 대해 설명한다. 3장은 본 연구를 통해 제안하고자 하는 P2P 기반 충돌예방 기법과 그를 위한 시스템 구조에 대해 소개하고, 4장에서는 시스템의 구현 결과를 실행 사례를 통해 기술한다. 5장에서는 기존 소프트웨어 협업개발 시스템과의 정성적인 평가를 통해 본 시스템의 유용성을 나타낸다. 마지막으로 6장에서 본 논문의 결론을 맺는다.

2. 관련 연구

2.1 동시성 제어기법

동시성 제어기법은 크게 낙관적(optimistic) 기법과 비관적(pessimistic) 기법으로 나뉜다[2]. 일반적으로 동시성 제어기법은 데이터베이스의 트랜잭션 처리 알고리즘에서 많이 다루어지는데, 데이터의 일관성 유지를 위해 비관적 기법은 각 트랜잭션 수행 시점에 검증을 수행하지만 낙관적 기법에서는 각 트랜잭션이 끝나는 시점에서만 검증을 수행한다. 대부분의 상용 데이터베이스 시스템이 locking protocol에 기반한 비관적 동시성 제어 기법을 사용하고 있음에도 불구하고 낙관적 동시성 제어 기법에 많은 관심을 보이고 있는 이유는 충분한 하드웨어 자원을 이용할 수 있다면 낙관적 동시성 제어 기법이 더욱 향상된 트랜잭션 처리 성능을 제공하기 때문이다[3, 11].

데이터베이스에서 데이터의 일관성 유지를 위해 적용하는 동시성 제어 기법은 협업개발 환경에서의 공유 자원의 충돌 문제를 해결하기 위한 것과 그 목적이 같음을 알 수 있다[4]. 전통적으로 많은 협업개발 지원도구들이 비관적 기법을 채택했는데 이는 지정 체크아웃(reserved checkout)을 이용한 배타적 잠금(exclusive locking) 기법에 기반한다. 배타적 잠금 기능을 이용하는 경우 동일 자원에 대한 동시 작업을 원천적으로 봉쇄함으로써 충돌을 방지한다. 이는 동일 자원에 대한 일관성을 보장하기 위한 확실한 방법이 될 수 있다. 하지만 이 기법은 특정 시점에 단 한 명의 사용자만이 자원을 수정할 수 있기 때문에 불필요한 작업 지연이 초래될 수 있다는 단점이 있다.

이와 반대로 낙관적 기법에서는 비 지정 체크아웃(unreserved checkout)을 이용한다. 이는 여러 사용자들이 자신의 로컬 복사본을 자유롭게 수정한 후, 체크인 하는 시점에서 그 충돌 여부를 판별하여 충돌을 해결(병합)할 수 있게 해주는 것으로 위의 비관적 기법과는 달리 공통 자원에 대한 사용자들의 동시 수정을 가능하게 한다[5]. 따라서 이 경우 불필요한 작업 지연이 초래되는 경우는 없다. 그러나 경우에 따라 사용자들의 충돌 인식 시점이 늦어지는 경우 충돌 해결에 대한 비용이 증가하고 동시 작업 중 중복된 작업이 진행될 수도 있다는 단점이 있다.

그 동안 몇몇 연구를 통해 비관적 기법에서의 일관성 확보와 낙관적 기법에서의 수행 성능을 고려한 동시성 제어기법들이 제안되었다. CIAO에서는 상호작용 성능을 위하여 준 낙관적 방식(semi-optimistic)을 이용한다[6]. 이는 공유 자원에 대해 소유권을 한 사람에게만 부여하고, 나머지 사

용자들은 그 자원의 복사본을 사용하게 한다. 소유권을 한 사람에게만 부여하는 것은 자원의 일관성을 유지하기 위한 비관적 기법의 모습이며, 복제된 자원을 다른 사용자들이 자유롭게 수정하도록 하는 것은 낙관적 기법에서 비롯된 것이라 볼 수 있다. 다만 복제 자원의 수정 결과를 반영하고 새로운 버전을 얻기 위해서는 반드시 소유권을 가진 사용자에 의해 검증을 받아야 한다.

유사한 예로 예측 기반의 동시성 제어 기법에 대한 연구가 있다[7]. 이 기법 또한 잠금 방식을 이용하여 데이터의 일관성을 유지한다. 다만 잠금을 요청할 사용자를 예측하여 그 사용자가 데이터를 수정하기 전에 미리 잠금을 주는 방식이다. 즉, 사용자가 잠금을 승인 받은 후 데이터를 수정하므로 비관적 방식처럼 완벽한 일관성을 제공할 수 있고, 사용자가 데이터를 수정하기 전에 잠금을 받기 때문에 잠금 응답에 필요한 지연 없이 바로 수정을 할 수 있으므로 낙관적 방식처럼 반응 시간이 빠르다. 하지만 예측 방법은 잠금을 미리 주고자 하는 사용자를 정확하게 예측해야 한다는 조건이 따른다. 만약 예측이 틀릴 경우는 예측 방식을 사용하지 않는 것보다 불필요한 시간 지연이 발생할 수 있다.

본 연구에서는 자원의 소유권을 특정 사용자에게 국한시키지 않고 모든 사용자가 자유롭게 그 복사본을 수정할 수 있도록 기회를 제공한다. 또한 공유 자원에 대한 소유권을 획득하기 위해 필요한 요청 및 응답 시간을 필요로 하지 않는다. 다만 복제의 대상이 되는 원래의 자원은 별도의 서버를 통해 따로 관리된다. 전통적인 낙관적 기법을 기반으로 하되 동시작업으로 인해 발생할 수 있는 충돌문제를 서버에 반영하는 시점이 아닌 충돌 발생 시점에 실시간으로 확인 및 해결할 수 있도록 하였다.

2.2 소프트웨어 협업개발 지원도구

인터넷의 시대를 맞아 소프트웨어 개발주기가 짧아지고 소프트웨어의 규모가 점차 확대되어 감에 따라 지리적으로 분산되어 있는 조직 간 효과적인 협업개발에 대한 요구가 증대되고 있다. 일반적으로 소프트웨어 협업개발 지원도구는 최근 불고 있는 오픈 소스 프로젝트(OSP, Open Source Project)의 개발 환경과 같이 지리적으로 분산되어 있는 개발 구성원들이 해당 사이트에서 소스코드 및 문서와 같은 주요 산출물에 대한 관리에서부터 프로젝트 관리, 릴리즈 관리 등 소프트웨어 개발 프로세스 전반에 필요한 모든 의사소통을 지원한다[13].

소프트웨어 협업개발 지원도구는 소스코드 관리, 이슈트래킹, 문서 관리, 풀 텍스트 검색, 패치 관리(patch management), 업무 관리(task management), SCM(Software Configuration Management) 도구와의 통합 기능 등과 같은 복합적인 기능들을 지원할 수 있어야 한다[8].

일반적으로 프로젝트 관리와 문서 관리 등 소프트웨어 프로세스 관리의 중요성이 높아지고 있으나, 이들 협업 지원 기능 중 소프트웨어 개발의 가장 핵심적인 산출물인 소프트웨어 소스코드에 대한 관리가 가장 중요성이 높은 것으로 꼽고 있다. 소스코드는 일반적인 요구사항 명세서나 설계 문서 등과 같은 산출물과는 달리, 복잡한 디렉토리 체계 내에 놓여 있으며 많은 파일 개수를 갖는 등의 특성을 지니고 있기 때문이다. 이러한 특성으로 인해 소스코드 관리 기능은 단순히 파일에 대한 버전관리 차원을 넘어 그 특성에 맞는 별도의 추가 기능을 가져야 한다. 그 중 대표적인 것으로 효과적인 차이점 비교 및 병합 기능, 워크스페이스(workspace) 관리 기능이 있다[9].

차이점 비교 및 병합은 ASCII 형식의 소스코드들에 대하여 버전 간 차이점을 비교하여 보여주고, 필요에 따라 상이한 버전 간 내용을 병합하는 기능이다. 워크스페이스 관리 기능은 로컬에서 수행되는 작업의 결과를 서버에 직접 반영할 수 있도록 하는 것이다. 소스코드의 특성 상 파일들은 원래의 디렉토리 구조 그대로 저장되어야 하는데 이를 위해 특정 디렉토리를 소스코드 관리도구에서 관리하는 특수 디렉토리로 지정하고 그곳을 워크스페이스로 지정한다. 즉, 워크스페이스란 여러 개의 소스파일이나 프로젝트 파일을 포함하는 하나의 작업공간이다. 워크스페이스 내에서는 최소한의 명령어 수행만으로 그 내부에서의 파일 변경 사항을 서버에 업로드/다운로드 할 수 있다.

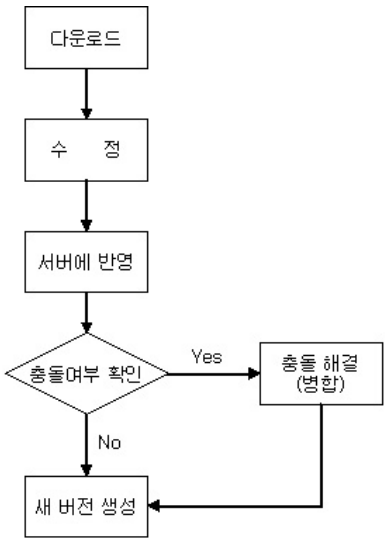
기존 도구들에서 제공하는 이러한 기능들 외에도 소스코드 관리를 위해서는 소스코드 개발 시 개발자 상호간의 의사소통을 지원하는 기능이 필수적으로 요구된다. 특히 소스코드의 경우 파일의 개수가 많고 그 파일들이 위치한 디렉토리 구조가 복잡한 경우가 많기 때문에 사용자들이 본인이 수정하는 파일 하나하나를 인식하여 서로에게 알리기 어렵다는 특성이 있다. 이 때문에 업무의 분업화가 잘 되어 있는 조직에서도 다수의 개발자가 동시에 같은 파일을 수정하는 상황이 빈번히 발생하며 협업개발 지원도구에서는 이를 효과적으로 제어하는 기법이 요구된다. 서론에서 밝힌 비관적, 낙관적 기법은 각각의 장점에도 불구하고 동시 수정 허용과 충돌 예방 두 가지 모두를 제공하지는 못한다. 이에 본 논문에서는 동시 수정을 허용하면서도 P2P기반 에이전트를 통해 충돌을 예방하는 P2P기반 충돌예방 기법을 제시한다.

3. P2P기반 충돌예방 기법

3.1 충돌예방 기법 개요

본 논문에서 제시하는 P2P기반 충돌예방 기법은 기본적으로 낙관적 기법을 근간으로 한다. 낙관적 기법은 소스코드 저장소(서버)에서 다운로드 받은 로컬 복사본의 자유로운 수정을 허용한다. 대신 수정한 파일에 대한 타 사용자와의 동시 수정(충돌) 여부는 서버에 업로드 하는 시점에서 서버가 판별하여 사용자에게 알린다. 이 과정은 (그림 1)과 같다.

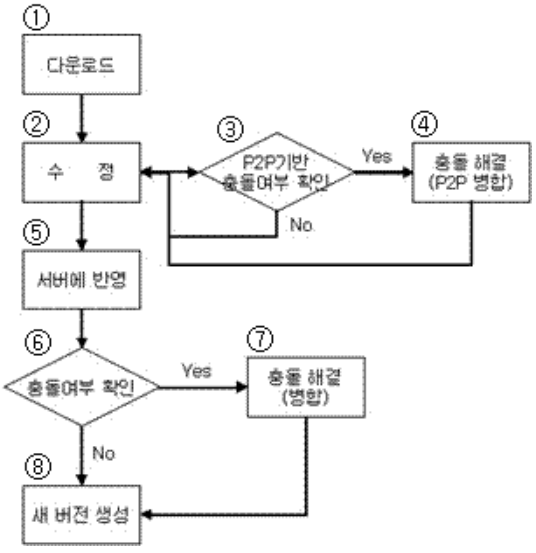
(그림 1)에서 보는 바와 같이 낙관적 기법의 단점은 다수의 공유 사용자들에 의해 동시 수정이 이루어지는 경우, 각각이 수정을 완료하고 서버에 반영하는 시점에야 그 충돌여



(그림 1) 낙관적 기법

부를 알 수 있다는 것이다. 만약 다운로드 한 자원의 수정을 완료하고 서버에 반영하고자 할 때 충돌 여부가 확인된다면, 다른 사용자에 의해 이미 업로드 된 새로운 버전을 서버로부터 다운로드 하여 차이점을 비교하고 병합하는 과정을 통해 충돌을 해결해야 한다. 이렇게 충돌 여부에 대한 확인이 늦어짐으로 인해 충돌을 해결(각각의 수정 내용을 하나의 버전으로 통합시키는 작업)하는 시점 또한 늦어지게 되므로 이는 불필요한 작업 지연의 이유가 될 수 있다.

이에 P2P기반 충돌예방 기법의 핵심은 (그림 2)와 같이 자원을 수정하는 시점에 그 충돌여부를 확인하고 해결할 수 있도록 하는 것이다. 즉, 충돌확인과 해결의 시점을 앞당겨 협업의 효율을 높이고자 하는 것이 본 논문의 핵심 취지라 할 수 있다. 이를 위해 본 논문에서는 네 개의 컴포넌트로 구성된 시스템 구조를 제안한다(그림 3). 각 컴포넌트의 역할을 간단히 살펴보면 다음과 같다.



(그림 2) P2P기반 충돌예방 기법

(1) 파일 시스템 변경 감지(File System Change Monitor) 컴포넌트
- 각 공유 사용자의 로컬 파일 시스템의 변경을 실시간으로 감지한다. 단, 감지 대상은 워크스페이스로 지정된 디렉토리 내 파일들이다.

(2) P2P 기반 상태 감시(P2P Status Observer) 컴포넌트
- 파일 시스템 변경 감지 컴포넌트로부터 전송된 파일 상태 감시 정보를 주기적으로 상대 공유 사용자들과 교환한다. 교환정보를 바탕으로 워크스페이스 내 각 파일의 충돌여부를 해석한다.

(3) P2P 기반 충돌 통보(P2P Conflict Notifier) 컴포넌트
- P2P 기반 상태 감시 컴포넌트로부터 충돌로 판별된 파일이 인식되는 경우 해당 정보(상대방 공유 사용자, 충돌이 발생한 파일 목록)를 사용자에게 통보한다.

(4) P2P 기반 충돌 해결(P2P Conflict Resolver) 컴포넌트
- 공유 사용자와 충돌이 발생한 경우, 충돌 해결에 필요한 파일들을 P2P로 전송 받아 이 파일들 간의 차이점을 비교하고 병합할 수 있도록 지원한다.

위의 컴포넌트들을 기반으로 (그림 2)에서 나타내는 과정을 설명하기 위해 다음과 같은 가정을 한다. 사용자A와 B는 서버로부터 자원 P를 다운로드 하여(단계 ①) 각자의 필요에 맞게 수정하고자 한다. 사용자 A가 자원 P를 수정하여 P'을 생성하게 되면 파일 시스템 변경 감지 컴포넌트는 이 사실을 감지하고 P2P 기반 상태 감시 컴포넌트를 통해 공유 사용자인 B에게 통보하게 된다. 사용자 B가 동일한 자원 P를 수정하여 P"을 생성하면(단계 ②) P2P 기반 상태 감시 컴포넌트에 의해 충돌 사실이 확인되고 해당 정보(상대방 사용자 및 충돌 대상 파일)는 P2P 기반 충돌 통보 컴포넌트에 의해 사용자 B에게 통보된다(단계 ③). 이 사실을 확인한 사용자 B는 P2P 기반 충돌 해결 컴포넌트를 이용하여 사용자 A에 의해 변경된 P'를 전송 받아 비주얼 차이점 비교를 실행할 수 있고, 통합된 결과를 얻기 위해 자신의 변경 결과와 병합할 수 있다(단계 ④). 비주얼 차이점 비교란 동일 자원에 대한 서로 다른 변경내용을 시각적으로 쉽게 파악할 수 있도록 하는 것이다. 4장의 (그림 7)에서 볼 수 있듯이 두 사용자가 동일한 자원에 대해 변경을 수행한 경우 변경이 이루어진 부분을 쉽게 찾을 수 있도록 하였다. 병합을 수행하는 과정에서 상대방 사용자와의 의견 교환이 필요하다면 메신저 기능을 이용하여 대화를 나눌 수도 있다.

단계 ④ 까지의 과정을 통해 P2P 기반의 충돌 해결이 완료되면 사용자는 통합된 결과를 서버에 반영하게 된다(단계 ⑤). 서버에 반영하는 시점에 다시 한 번 서버의 최신버전과 충돌여부를 확인하고(단계 ⑥) 충돌이 발견되지 않으면 반영이 완료되어 새로운 버전이 생성된다(단계 ⑧). 만약 서버에 반영하기 전에 이미 다른 사용자에 의해 새로운 버전이 생성된 경우, 기존 낙관적 기법에서처럼 서버의 새 버전과 자신이 수정한 내용을 병합해야 한다(단계 ⑦). 이는 공유

사용자들 간 충돌 확인이 제 때 이루어지지 않거나 서버에 반영하는 시점의 차이로 인해 발생하는 경우로, 충돌 예방 기법의 예외적인 상황이라 볼 수 있다.

3.2 시스템 구조

서론에서 제시한 P2P기반 충돌예방의 세가지 핵심 기능을 위해 (그림 3)의 시스템 구조를 제시한다. 각 사용자의 로컬 시스템에는 통합 워크스페이스 환경과 함께 3.1절에서 소개한 네 개의 컴포넌트들이 설치된다. 이를 바탕으로 공유 사용자들 간에 P2P 통신이 이루어지고, Client/Server 구조를 통해 필요한 자원을 다운로드 하거나 수정된 결과를 반영할 수 있다.

본 절에서는 (그림 3)의 시스템 구조 중 3.1절에서 설명한 네 개의 컴포넌트들에 대해 그 동작 원리를 자세히 설명한다.

3.2.1 파일시스템 변경 감지(File System Change Monitor) 컴포넌트

파일 시스템 변경 감지 컴포넌트는 워크스페이스로 지정된 디렉토리 내 파일들의 변경을 실시간으로 감지하는 기능을 수행한다. 파일의 변경이 감지되면, 이는P2P 상태 감시 컴포넌트의 전송대기 큐(queue)에 쌓여 본인의 워크스페이스 상태를 공유 사용자들에게 전송할 준비를 하게 된다. 그런 다음 정해진 주기에 맞추어 공유 사용자들 간 파일의 변화 상태를 P2P로 교환하게 되는데, 이 때 동일한 워크스페이스 내의 동일한 파일을 수정한 경우, 충돌이 발생하였음을 각각의 공유 사용자에게 알리게 되는 것이다.

3.2.2 P2P기반 상태 감시(P2P Status Observer) 컴포넌트

P2P기반 상태 감시 컴포넌트는 파일 시스템 변경 감지 컴포넌트로부터 전송된 파일 상태 감시 정보를 전송 대기 큐에 쌓아두었다가 주기적으로 상대 공유 사용자들과 교환하는 기능을 수행한다. 이러한 교환 정보를 바탕으로 워크스페이스 내 각 파일의 충돌 여부를 해석한다.

Windows NT/2000/XP 이상 사용자의 경우 상태 감시 컴포넌트를 윈도우 서비스로 등록하면, 사용자가 로그아웃을 한 경우에도 다른 공유 사용자의 상태 정보 교환 요청에 응답할 수 있다는 장점을 지니고 있다.

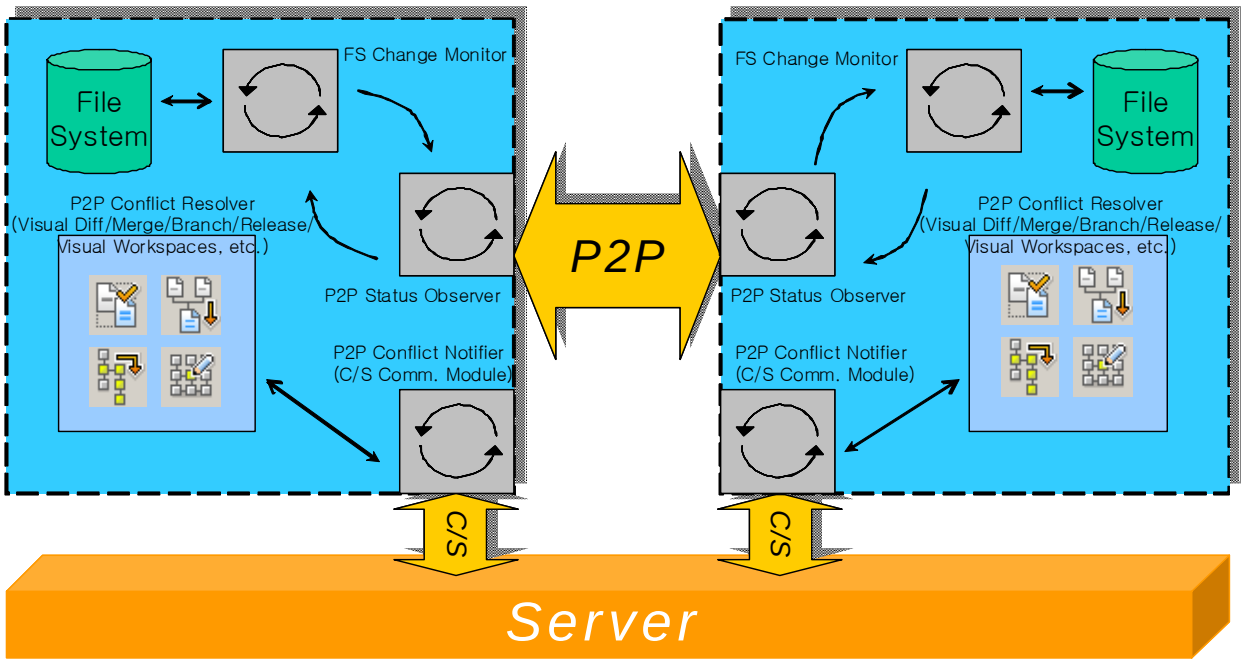
3.2.3 P2P기반 충돌 통보(P2P Conflict Notifier) 컴포넌트

P2P기반 충돌 통보 컴포넌트는 최근 많이 활용되고 있는 인스턴트 메신저의 메시지 도착 알림 기능과 유사하다. P2P 기반 상태 감시 컴포넌트로부터 충돌로 판별된 파일이 인식되는 경우, 충돌이 발생한 공유 사용자 및 충돌이 발생한 파일의 목록을 사용자에게 알리며 이를 해결할 수 있는 컴포넌트를 직접 호출하거나 해당 파일과 연결된 전용 편집기(텍스트 편집기, IDE 등)를 실행할 수 있도록 해준다.

본 컴포넌트는 윈도우 트레이에 항상 실행 중 상태로 되어 있어 다른 작업 중에도 손쉽게 충돌 여부를 확인할 수 있도록 한다.

3.2.4 P2P기반 충돌 해결(P2P Conflict Resolver) 컴포넌트

P2P 기반 충돌 해결 컴포넌트는 비주얼 차이점 비교 및 병합 기능을 P2P 기능에 통합한 것이다. 충돌이 발생한 경우, 충돌 해결에 필요한 파일들을 P2P로 전송 받아 이 파일들 간의 차이점을 비교하고 이를 병합할 수 있는 기능이 요



(그림 3) 시스템 구조

구된다. 대체로 소스코드 관리에 많이 활용되므로 소스코드 편집기 등에서 지원하는 대부분의 기능들(syntax highlighting, undo/redo, copy/paste/cut, print)을 제공해야 하며, 차이점을 효과적으로 표시해주고 이 정보를 바탕으로 사용자들이 정확하게 충돌을 해결할 수 있도록 지원하여야 한다.

4. 시스템 구현

제시한 시스템은 Beeskit standard/collaborative edition이라 부르며 이는 협업개발을 지원하기 위한 도구이다. Beeskit은 위의 협업개발 지원도구와 함께 형상관리 업무의 통합을 위한 integrated edition을 포함한다[17].

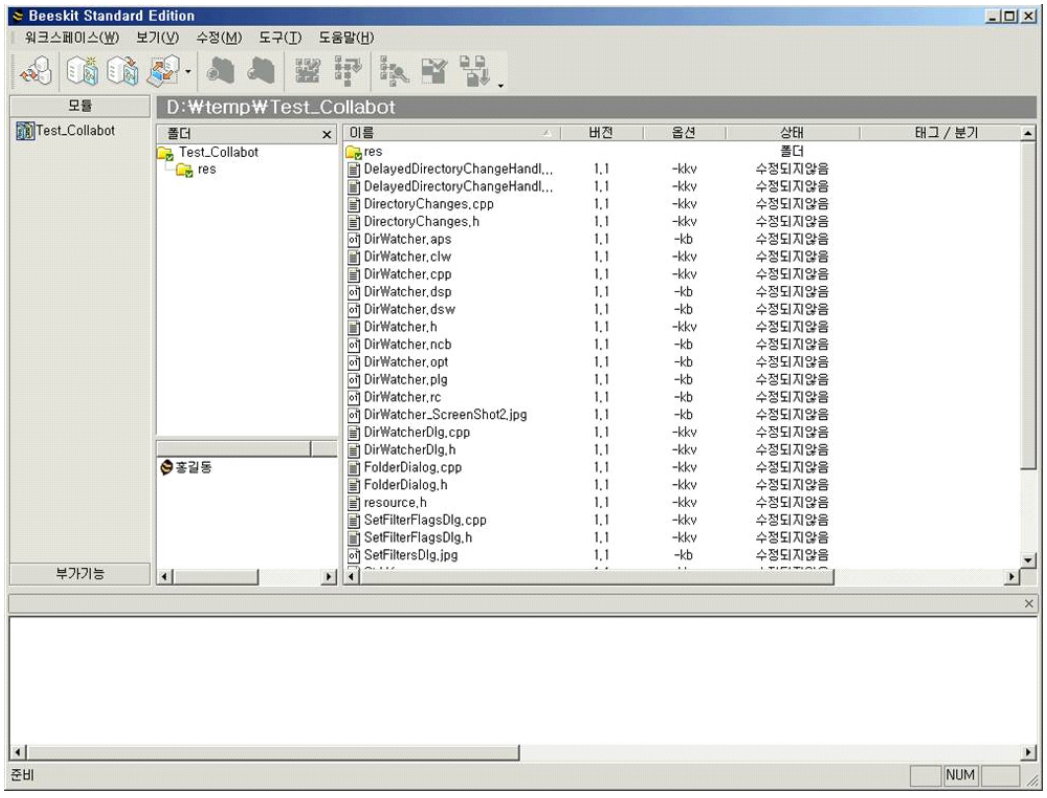
본 연구의 대상인standard/collaborative edition은 CVS의 버전제어 기법과 저장소 관리 메커니즘을 기반으로 하였으며 C++/MFC 기반으로 구현되었다. 단순히 개발 산출물에 대한 버전관리뿐만 아니라 효율적인 병렬개발 및 동시성 제어를 지원하기 위해 P2P 기반으로 구현하였다. 본 시스템은 Windows 환경을 기반으로 개발되었으며 자원에 대한 획득 및 수정의 권한을 부여하기 위해 access control list를 관리한다. 이 정보는 중앙 서버에 저장되며 공유 사용자들간의 유기적인 통신을 위해서도 사용된다.

3.2절에서 소개한 시스템 구조는 클라이언트들 간 충돌해결을 위한 P2P 구조와 공유자원에 대한 작업의 결과를 저장소(서버)에 반영하기 위한 Client/Server 구조의 조합된 모습을 보여주고 있다. 또한 앞서 언급한 4개의 컴포넌트들은 각 클라이언트 시스템에서 운용된다.

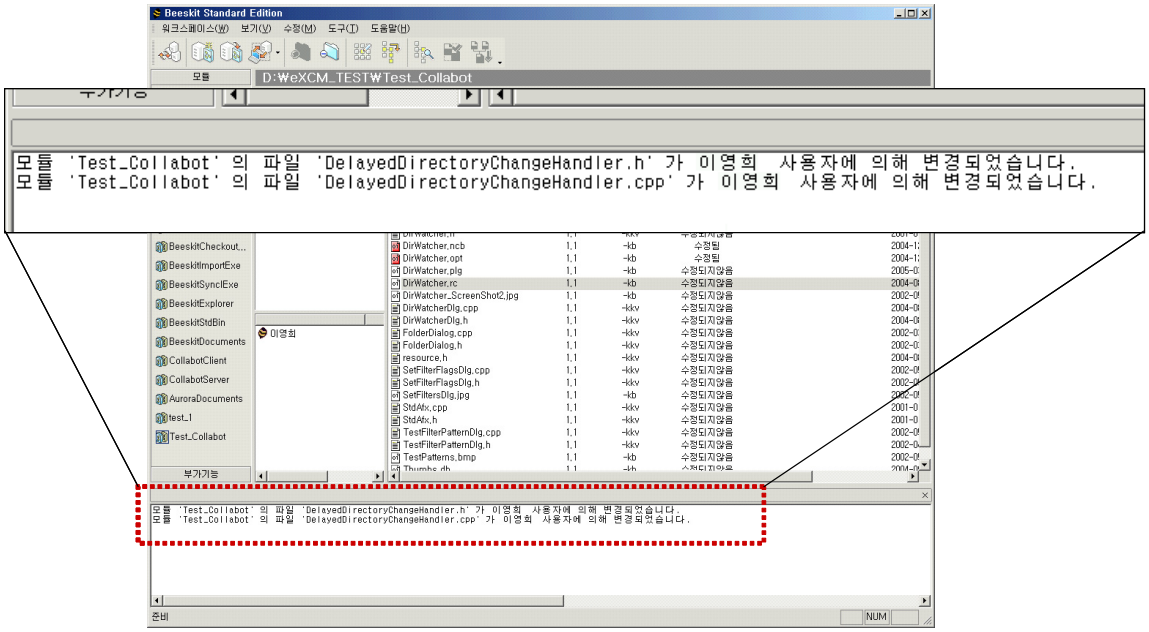
본 연구에서 다루는 P2P 기반 협업개발 지원도구는 대부분의 전통적인 협업개발 지원도구에서처럼 배타적 잠금을 이용하지 않고 여러 명의 사용자가 동시에 동일 자원을 체크아웃 받을 수 있도록 한다. 여기서 체크아웃이란 저장소(서버)에 의해 관리되는 기본 단위인 모듈을 자신의 로컬에 다운로드 하는 것을 의미한다. 모듈이 각 사용자의 로컬에 저장되면 이것은 하나의 워크스페이스가 되는 것이다.

체크아웃의 개념은 기존의 CVS에 기반한 것으로서, CVS는 뒤에서 다룰 버전관리나 차이점 비교, 병합 등의 기능에 대한 근간을 제공하는 공개시스템이다[10]. 일단 여러 명의 사용자에 의해 체크아웃 된 모듈은 그들의 공유 대상이 되며, 공유하고 있는 사용자들은 시스템의 사용자 목록을 통해 확인할 수 있다. (그림 4)는 사용자가 시스템에 로그인한 후 특정 모듈을 체크아웃 한 화면이다((그림 2)의 단계 ①에 해당). 체크아웃 된 모듈에는 여러 개의 파일들이 존재하며 이들 각각이 실제 수정의 대상이 된다. 또한 자신이 체크아웃 한 모듈에 대해서 공유하고 있는 다른 사용자들의 목록을 확인할 수 있다. 이러한 워크스페이스는 차후 모든 변경이 완료된 후 서버에 손쉽게 체크인할 수 있도록 효율적인 GUI 환경을 제공하고 있다.

(그림 4)에서 보는 바와 같이 여러 사용자들에 의해 공유되는 모듈에 대해서 시스템은 그 변경사항(파일 수정, 삭제)을 감시하게 된다. 모듈 내의 특정 파일들이 변경되면 시스템은 해당 내용을 인식하고 다른 P2P 공유 사용자들에게



(그림 4) 시스템 로그인 및 모듈 체크아웃 결과화면

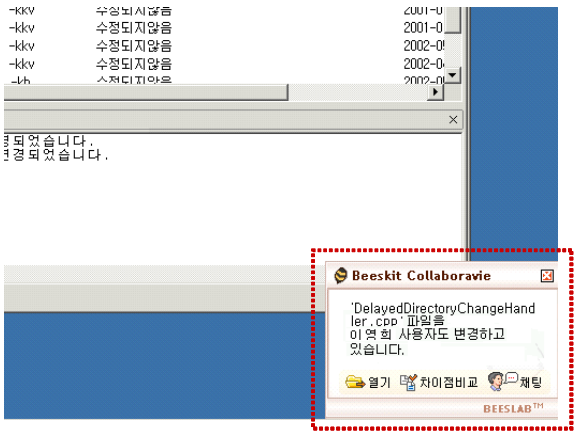


(그림 5) 파일변경 통보 메시지 수신

메시지를 통보한다.(그림 5)는 다른 사용자에 의해 공유되고 있는 동일 모듈의 특정 파일을 수정((그림 2)의 ②에 해당)했을 경우 충돌여부를 확인하고 통보해주는 모습을 나타낸다.

동일 모듈을 공유하고 있는 다른 사용자들은 언제든 그 안에 포함된 같은 파일을 수정할 수 있다. 이러한 경우 앞서 언급한 충돌이 발생하게 되는데, 시스템은 같은 파일의 수정으로 인한 충돌 사건을 당사자들에게 통보((그림 2)의 ③에 해당)해야 한다(그림 6).

윈도우의 트레이를 통해서 다른 사용자와의 충돌통보를 받게 되면 충돌이 일어난 파일에 대해서 차이점 비교를 실행할 수 있다(그림 7). 차이점 비교를 통해 자신의 변경 내용과 상대방의 변경 내용을 병합((그림 2)의 ④에 해당)함으로써 통합된 결과를 서버에 반영할 수 있다. 부가적으로 앞서 수정한 사용자와 변경관련 내용에 대해 상의할 수 있는 의사소통의 역할 또한 제공해야 한다. (그림 6)의 트레이에

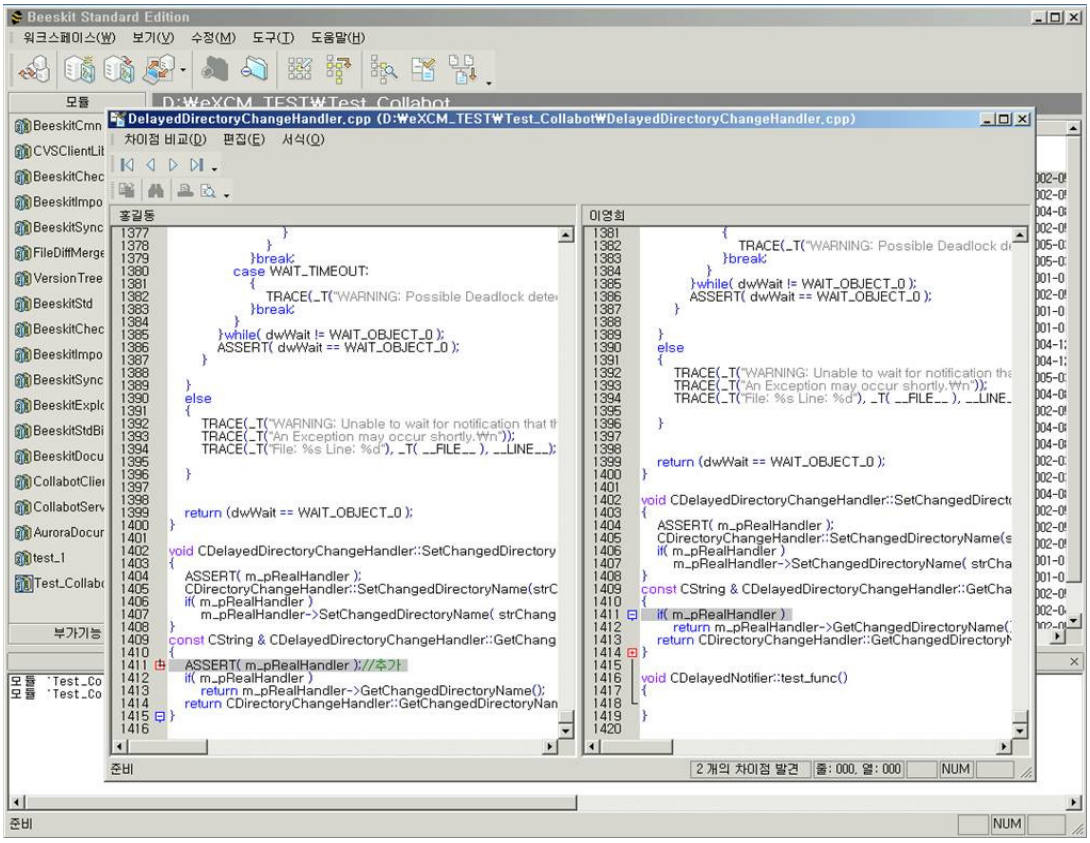


(그림 6) 충돌통보 메시지 수신 화면

서 채팅 메뉴를 선택하면 메신저를 통해 상대방과 관련 내용에 대한 의견을 교환할 수 있다.

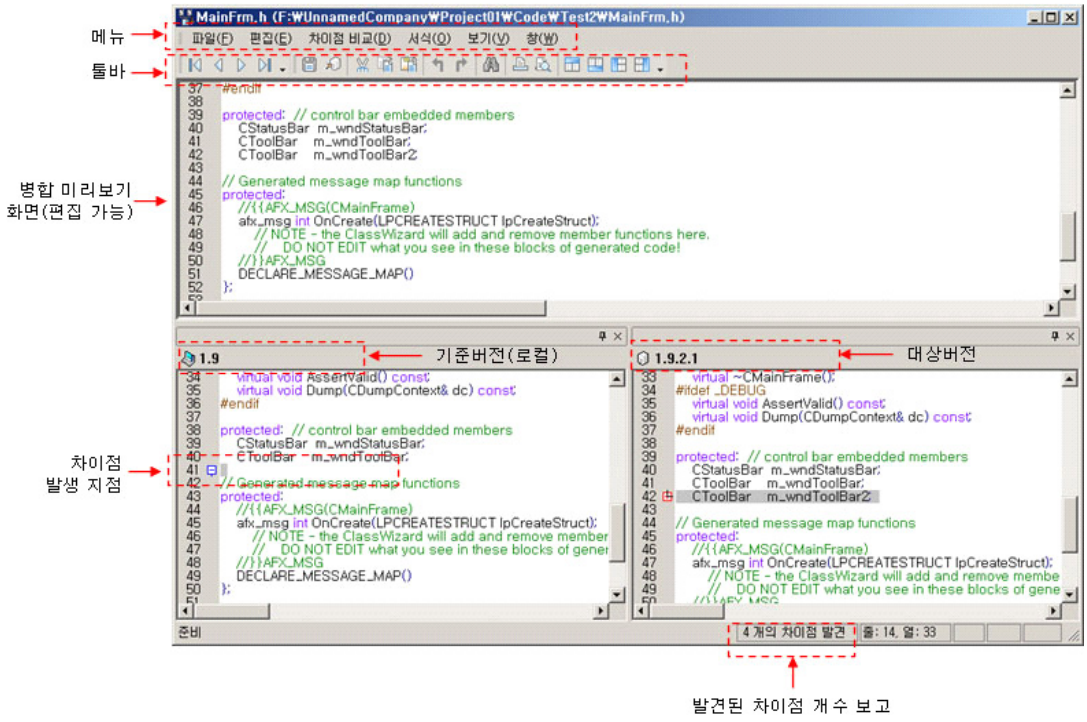
이렇게 변경된 파일을 서버에 반영(commit)하여((그림 2)의 단계 ⑤에 해당) 새로운 버전이 생성된 경우에는, 곧바로 공유 사용자들에게 메시지를 통보하게 되며, 메시지를 받음과 동시에 공유 사용자들에게는 서버에 반영 통보 메시지가 표시된다. 파일 리스트에도 새로운 버전이 생성되었다는 정보를 표시해준다. 사용자는 파일의 상태에 따라 "단순 업데이트", "병합" 등의 작업을 선택적으로 수행할 수 있다. 사용자의 설정 여부에 따라 새로운 버전이 자동으로 업데이트하는 작업을 수행할 수도 있다.

(그림 2)의 흐름도는 서버에 반영하는 작업으로부터 새 버전이 생성되기까지의 과정이 기존 낙관적 기법 (그림 1)과 크게 다르지 않음을 확인할 수 있다. 즉, 서버에 반영하는 과정에서 발생하는 충돌 문제와 이를 해결하기 위한 노력이 그대로 존재한다. 이는 본 연구의 취지에 어긋나는 것으로 보일 수 있으나, 충돌의 문제를 P2P환경에서 모두 해결하지 못할 수도 있다는 가정을 전제로 한다. 예를 들어 자신이 변경한 내용에 대해 충돌의 원인을 제공한 사용자가 병합된 내용에 대한 협의 없이 먼저 서버에 반영한다면 병합을 수행한 사용자가 서버에 반영하는 시점에서는 또다시 서버의 내용과 충돌이 일어날 수 있다((그림 2)의 ⑥에 의해 확인). 즉, 서버로부터 다운로드 한 시점의 파일에 대해서 새로운 버전이 생성된 경우로 해석할 수 있을 것이다. 이러한 경우 서버에 반영하는 과정에서도 P2P기반의 차이점 비교 및 병합과 같은 기능을 통해서 최종적으로 충돌문제를 해결((그림 2의 ⑦에 해당)할 수 있다. 다만 비교 및 병합의 대상이 다른 사용자의 로컬에 존재하는 파일이 아닌 서버에 이미 반영된 파일이라는 것이 다르다. 따라서 비교 및 병합의 대상에 대해서 특정 버전을 선택할 수 있어야 한다. 만



(그림 7) P2P 공유 사용자와의 차이점 비교

약 위에 든 예와 같은 상황이라면 서버 내 최신 버전과의 비교를 통해 병합을 수행(충돌 해결)해야 할 것이다. (그림 8)은 서버 내의 파일과 비교를 통해 병합할 수 있는 화면을 보여준다.



(그림 8) 서버 내 파일과의 병합화면

5. 유용성 평가

본 논문에서 제시한 P2P기반 지능형 에이전트는 소프트웨어 개발 시 충돌예방을 통해 효율적인 협업환경을 지원하기 위한 도구이다. 본 절에서는 소프트웨어 협업개발을 지원하는 여러 유명 제품들과의 정성적 비교평가를 통해 본 논문에서 제시한 시스템의 기능적 유용성을 설명하고자 한다.

평가의 기준이 되는 항목은 협업개발 환경의 핵심인 병렬개발 및 분산환경의 지원 여부를 비롯해 도구의 사용 용이성에 초점을 두었다[15,16]. <표 1>은 본 절에서 평가의 기준이 되는 항목들에 대해 간단히 설명하고 있다.

Graphical checkin tool은 수정한 파일을 서버에 반영하고자 할 때 복잡한 명령어의 수행 없이 간편하게 해당 기능을 지원할 수 있어야 함을 의미한다.

Dynamic branching 기능은 보다 효율적인 병렬개발을 지원하기 위해 제공되는 것으로 본문에서 설명하고 있는 동시작업의 흐름은 기본적으로 하나의 메인 흐름에 의해 진행되는 업무를 가정하고 있다. 분기의 생성(branching)은 서로 다른 개발흐름을 생성하여 병렬개발을 진행하고자 할 때 많이 활용된다. 버그 수정, 기능 개선 등을 병렬적으로 진행하여야 하는 경우가 그 예가 될 수 있다. (그림 9)는 분기의 생성과 병합의 과정을 보여주는 버전 그래프이다.

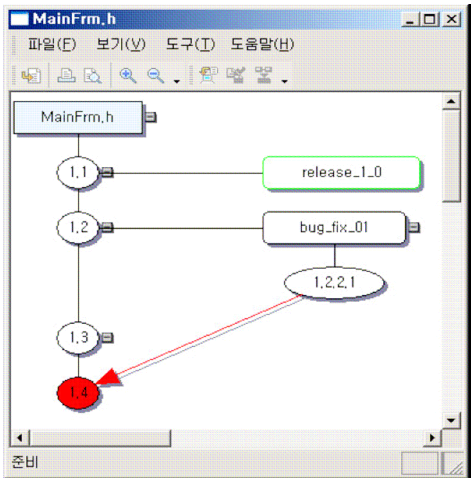
Peer-to-Peer architecture는 분산된 환경의 자원 공유와 작업 상황에 대한 정보의 교류가 서버를 통해 이루어져야 하는 일반적인 시스템들의 한계를 해결하기 위한 목적이다.

True parallel development는 Dynamic branching 항목과 더불어 실시간으로 여러 명의 동시 개발을 지원하고 있는지를 평가하기 위한 항목이다.

Multi-site development는 특정 영역에 국한되지 않고 어느 곳에서든(인터넷으로 연결된 환경) 필요한 자원의 획득과 병렬 개발의 가능성을 제공하는가에 대한 여부를 묻는 항목이다.

Post-event trigger는 특정 업무의 결과나 다른 사용자의 작업 상황에 따라 향후 어떤 일을 수행해야 하는지를 판단할 수 있도록 지원하는지 평가하기 위한 항목이다.

Accurate recording of all history는 개발이 진행되면서 특정 자원의 변경과 관련한 모든 기록을 유지하고 제공할 수 있는가를 판단하기 위한 항목이다. 시스템은 누가 언제



(그림 9) 버전 그래프(분기 생성 및 병합)

어떠한 목적으로 작업을 수행했는지 기록할 수 있어야 하며 이는 다수의 사용자가 참여하는 병렬개발에 있어 보다 정확한 업무를 수행할 수 있도록 하는 정보로 활용될 수 있다. 특정 자원에 대한 반복된 변경은 여러 개의 버전을 생성하게 되므로, 버전들 사이의 변경 이력(change history)을 파악하는 것 또한 'Accurate recording of all history' 항목에 속한 내용이라 볼 수 있다. 버전들 사이의 차이점 비교나 (그림 9)의 버전그래프에서 나타난 것처럼 특정 버전에 tag('release 1.0'은 버전 '1.1'의 tag)를 부착하는 것은 변경의 과정 및 의도를 파악할 수 있도록 도와준다. 이는 변경에 의한 자원의 추적성(traceability)을 제공하고 결과적으로 모든 변경 업무에 대해 정확한 기록을 제공할 수 있게 된다. 자원에 대한 추적성은 자원의 변화과정을 기술하고 이해할 수 있는 능력으로 정의되는데[18], 앞서 소개한 비주얼 차이점 비교 기능이나 tagging 기능은 자원의 변화과정을 나타내고 판단하는데 유용하게 사용될 수 있다.

ClearCase를 제외한 다른 도구들은 분기(branching)에 대한 한계를 갖고 있다. CVS, Subversion, VSS는 모두 단일 저장소 모델을 사용한다. 모든 사용자는 동일한 저장공간을 사용하며 만약 누군가 메인 트리의 변화를 주었다면 이는 모든 사용자들에게 그 영향을 주게 된다. 즉, 원본에 대한 컨트롤만을 허용하기 때문에 revision 컨트롤에 대한 여지가 존재하지 않는다. 따라서 병렬 개발을 수행할 경우 매번 정

〈표 1〉 유용성 비교를 위한 정성적 평가기준

항 목	설 명
Graphical checkin tool	편리한 GUI 환경을 통한 checkin 실행의 사용 용이성 제공
Dynamic branching	워크스페이스는 언제든지 새로운 분기로 개발될 수 있어야 하며, 분기의 실행에 대한 고도의 계획이 필요 없음
Peer-to-Peer architecture	Open source 스타일의 자원공유를 통해 빠른 개발속도를 제공
True parallel development	병렬개발에 의한 생산성의 향상과 빠른 시장 진입
Multi-site development	분산 환경에서의 개발을 지원. 지리적 한계를 극복하기 위함
Post-event triggers	특정 사건과 그 다음으로 해야 할 업무에 대한 통보기능. 보다 쉽고 효율적인 프로세스 관리를 제공
Accurate recording of all history	누가 언제 무슨 일을 했는지 쉽게 파악할 수 있도록 함으로써 병렬개발의 완성도를 높여주기 위한 기능

보의 손실을 겪게 되는데, 왜냐하면 누군가 이미 체크인 한 결과에 누군가 또다시 새로운 변경 내용을 체크인 하려고 한다면 기존 내용에 대해 병합작업을 수행해야 하고 이는 이전 작업결과의 손실을 야기하기 때문이다. 즉, 병합 이전의 모든 워크스페이스의 상태는 영원히 잃게 되는 것이다. 이를 달리 설명하면 N개의 병렬개발이 이루어지는 경우 마지막 반영 내용을 제외한 N-1개의 반영 결과는 없어지게 되는 것이다. 이러한 이유로 CVS, Subversion, VSS의 경우 분기에 의한 작업은 거의 불가능한 것으로 여겨지고 있다.

ClearCase는 전형적인 중앙 집중형의 시스템 구조를 가지고 있다. 서버가 설치된 로컬 영역에 대해서만 클라이언트의 접근을 허용한다. 이는 분산 환경에서의 접근성을 제공하지 못한다는 한계를 지니고 있다. 이러한 문제를 극복하기 위해서는 별도로 고비용의 제품 패키지를 구매해야 하며 이는 사용자들에게 적지 않은 부담으로 작용할 수 있다. 더구나 분산 환경의 지원을 위해 적용된 이 방침도 완벽한 기능을 제공하지 못하고 있다. ClearCase에서 분산 환경을 지원하기 위해 적용한 아이디어는 각 사이트에 분기(branch)를 할당하는 것이다. 하지만 ClearCase의 분기는 본 연구의 Dynamic branching의 경우와는 다르다. 여기서 제공된 분기는 오로지 한 사이트에서만 수정이 가능하다. 즉, 다른 사이트의 분기는 단지 읽기만 가능하다. 이는 서론에서 언급한 비관적 기법의 예로 볼 수 있다. 본 연구의 시스템이 제시한 dynamic branching은 다수의 사용자가 동시에 각자의 분기를 생성하여 읽기뿐만 아니라 수정까지도 가능하게 하는것을 의미한다면 ClearCase 의 branching은 자원의 소유권을 가진 사용자 외에는 수정의 권한이 부여되지 않음을 의미하는 것으로 근본적인 병렬개발의 목적에 한계를 지니고 있음을 알 수 있다. 결론적으로CVS, Subversion, VSS에서 사용한 단일 저장소 모델이나 ClearCase의 중앙 집중형 시스템 구조는 분산된 사용자들에게 적절한 병렬개발 환경을 제공하지 못하고 있음을 알 수 있다.

Peer-to-Peer(P2P) 아키텍처는 작업하고자 하는 자원의 공유를 위한 통신 구조이다. 앞서 설명한 다른 도구들과는 달리 서버에 의해 관리되는 master 자원의 복사본을 모든 개발자들이 자유롭게 수정할 수 있도록 지원하며 본 논문에서 제안하는 충돌 예방 및 해결 알고리즘을 지원하기 위한 구조이다. 이러한 구조를 통해 다양한 형태의 병렬개발이 가능해지며, 또한 분산 환경에서도 공간의 한계를 벗어날 수 있다.

비교항목의Dynamic branching이나Peer-to-Peer architecture, True parallel development, Multi-site development 항목은 동일한 관점으로 해석할 수 있다.

Post-event trigger는 개발자가 수행하는 작업의 흐름을 원활히 할 수 있도록 하는 것으로 충돌 발생 시 그에 대한 사건의 통보와 차이점 비교 등의 기능을 수행할 수 있도록 한 것이 그 예가 될 수 있다. Accurate recording of all history는 보다 완벽한 병렬개발 환경을 지원하기 위한 것으로, 모든 개발자들의 작업이력을 기록하고 보고하는 기능이다. P2P 환경에서 각 클라이언트는 특정 자원과 관련된 모든 개발자들의 작업 내용을 실시간으로 감지할 수 있으며, 사용자들의 등록정보를 관리하는 서버는 이러한 모든 사건에 대해 로그(Log) 기록을 저장함으로써 누가 언제 무슨 작업을 수행했는지 조회할 수 있도록 한다. VSS의 경우 동일 자원의 새로운 버전에 대해 충분한 배경이나 목적을 기록하지 않으므로 병렬개발에 있어 다른 사용자들의 변경 의도를 파악하기 어렵다는 한계가 있다.

<표 1>의 항목을 기준으로 다른 유명 협업관리 도구들과의 기능적 유용성을 평가한 결과가 <표 2>에 나타나 있다.

본 연구에서 제시한 시스템은 CVS의 버전제어 기능을 기반으로 자원의 버전을 관리하였다. CVS 서버를 통해 관리하고자 하는 자원을 저장하고 자원의 획득 및 수정권한을 사용자들에게 부여하는 등의 기능이 구현되었다. 하지만 기능의 사용 용이성을 위한 GUI 기반의 환경, 단순 버전기법이 아닌 분기를 이용한 버전관리, 이를 통한 효과적인 병렬개발, 분산 환경에서의 자원공유 및 의사소통을 지원하기 위한 P2P 기반의 시스템 구조, 업무의 흐름을 파악하고 다음 해야 할 업무를 수행할 수 있도록 하는 사건통보 기능, 업무와 특정 사건의 결과에 대한 기록 및 유지와 같은 기능은 분명 CVS에서 충분히 제공하지 못하는 부분이다.

6. 결 론

최근 소프트웨어의 종류와 규모가 날로 증가함에 따라 개발 과정에서 발생할 수 있는 여러 가지 시행착오를 줄이고 높은 수준의 품질관리를 위해서는 효과적인 협업개발 지원 도구의 필요가 절실하다. 그러나 아직까지 국내의 소프트웨

〈표 2〉 유명 협업관리 도구들과의 기능성 비교

	CVS	Subversion	ClearCase	VSS	Beeskit
Graphical checkin tool	No	No	Yes	Limited	Yes
Dynamic branching	No	No	No	No	Yes
Peer-to-Peer architecture	No	No	No	No	Yes
True parallel development	No	No	Yes(complex)	No	Yes
Multi-site development	No	No	Yes(add on)	No	Yes
Post-event triggers	Weak	Yes	Yes	No	Yes
Accurate recording of all history	No	No	No	No	Yes

어 개발업체들은 몇몇 대규모 조직을 제외하고는 효율적인 협업환경을 구축하지 못하고 있는 것이 현실이다.

본 연구에서는 소프트웨어를 개발함에 있어 조직 단위의 협업개발을 원활하게 수행할 수 있도록 지원하는 P2P 기반의 협업개발 지원도구를 제시하였다. P2P 기반의 충돌예방 및 해결 기법은 여러 개발자들의 동시작업에 의한 자원의 일관성 을 유지함과 동시에 기존 낙관적 기법이 가지는 단점을 극복할 수 있는 방안으로 제시되었다. 본 연구를 통해 개발된 시스템은 비교적 대중화되어 있는 버전관리 시스템인 CVS의 기능적 요소들을 흡수하여 보다 높은 수준의 소프트웨어 협업개발 환경 구축을 지원하는 것을 목표로 하고 있다. 여기에 branch/merge 모델을 적용한 확장된 버전관리 기능을 제공함으로써 소프트웨어 개발의 유연성을 높이고자 하였다.

나아가 소프트웨어 개발 시 프로젝트 관리, 프로세스 관리, 변경 관리 기능 등을 지원하는 형상관리 시스템과의 통합 환경을 구축함으로써 협업개발 전반의 업무 효율성을 높이고자 노력하고 있다. 이를 통해 개발 조직의 성숙도를 높이고, 국내 소프트웨어의 품질 향상에 이바지 하는 것이 향후 연구의 목적이다.

참 고 문 헌

[1] Joanna DeFranco-Tommarello, Fadi P. Deek, “Collaborative Software Development: A Discussion of Problem Solving Models and Groupware Technologies”, Proceedings of the 35th Hawaii International Conference on System Sciences 2002.

[2] A. Makni, R. Bouaziz, F. Gargouri, “Formal Verification of an Optimistic Concurrency Control Algorithm using SPIN”, Proceedings of the 13th International Symposium on Temporal Representation and Reasoning, pp.160-167, June. 2006.

[3] 최희영, 황부현, “중복 데이터베이스 시스템에서 낙관적인 원자적 방송을 이용한 동시성제어 기법”, 정보처리학회논문지 D 제8-D권 제5호, Oct., 2001.

[4] R. Strom, G. Banavar, K. Miller, A. Prakash, M. Ward, “Concurrency Control and View Notification Algorithms for Collaborative Replicated Objects”, IEEE Transactions on Computers, Vol.47, No.4, pp.458-471, April, 1998.

[5] Haifeng Shen, Suiping Zhou, Chengzheng Sun, “Flexible Concurrency Control for Collaborative Office Systems”, 3rd International Conference on Information Technology and Applications, Vol.2, pp.45-50, July, 2005.

[6] Sung. U, Yang. J, Wohn. K, “Concurrency Control in CIAO”, IEEE Computer Society, 1999.

[7] 양정화, 이동만, “분산 협동 환경을 위한 객체 중심 동시성 제어 기법”, 한국정보과학회 가을 학술발표 논문집 Vol.26, No.2, 1999.

[8] Melissa Webster, “An End-User View of the Collaborative Software Development Market”, IDC, Dec. 2003.

[9] Ovum, “Ovum Evaluates: Configuration Management”, Ovum, 2003.

[10] Andrew Hunt, Dave Thomas, “실용주의 프로그래머를 위한 버전관리 using CVS”, Insight, 2004.

[11] Li Gouhui, Yang Bing, Chen Jixiong, “Efficient Optimistic Concurrency Control for Mobile Real-Time Transaction in a Wireless Data Broadcast Environment”, Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pp.443-446, Aug, 2005.

[12] Min Li, Shuming Gao, Jerry Y. H. Fuh, Yunfeng Zhang, “A Fine Granular Concurrency Control Mechanism for a Peer-to-Peer Cooperative Design Environment”, 11th International Conference on Computer Supported Cooperative Work in Design, pp.180-185, April, 2007.

[13] Joanna DeFranco-Tommarello, Fadi P. Deek, “Collaborative Software Development: A Discussion of Problem Solving Models and Groupware Technologies”, Proceedings of the 35th Annual Hawaii International Conference on System Sciences, pp.568-577, Jan, 2002.

[14] Min Tang, Shang-Ching Chou, Jin-Xiang Dong, “Concurrency Conflict Solving for Collaborative Feature Modeling”, Proceedings of the 9th International Conference on Computer Supported Cooperative Work in Design, Vol.1, pp.50-55, May, 2005.

[15] http://www.bitkeeper.com/Comparisons.html

[16] http://better-scm.berlios.de/comparison/comparison.html

[17] “P2P 기반 소프트웨어 협업개발 지원도구 기술, Beeskit”, 우수신기술 지정・지원사업 최종보고서, 2005.

[18] Mohan, K. and Jain, R., “Using Traceability to Mitigate Cognitive Biases in Software Development”, Communications of the ACM, Vol.51, No.9, pp.110-114, Sep., 2008.

박 현 수

e-mail : hspark@bcc.ac.kr

1985년 경북대학교 전자과(학사)

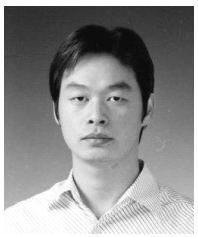
1988년 경북대학교 전자계산학과(공학석사)

1998년 충남대학교 전자계산학과(이학박사 수료)

1990년~1998년 한국전자통신연구원 선임연구원

1999년~현 재 백석문화대학 컴퓨터정보학부 부교수

관심분야 : 소프트웨어공학, 소프트웨어 아키텍처, 소프트웨어 형상관리



김 대 엽

e-mail : kdymn2@cnu.ac.kr

2005년 충남대학교 정보통신공학부(학사)

2005년~현 재 충남대학교 컴퓨터공학과 석·박사 통합과정

관심분야 : 소프트웨어 형상관리, 추적성 관리, 컴포넌트 기반 개발방법론

윤 청

e-mail : cyoun@cnu.ac.kr

1979년 서울대학교 물리학과(학사)

1983년 Illinois State University 전산학과(석사)

1988년 Northwestern University 전산학과(박사)

1983년~1985년 Wayne State University 전산학과 전임강사

1985년~1987년 Northwestern University 전산학과 전임강사

1988년~1993년 Bell Communications Research 선임연구원

1993년~현 재 충남대학교 전기정보통신공학부 교수

관심분야 : 소프트웨어공학, 객체지향 개발방법론