워크스페이스 기반 소프트웨어 형상관리 시스템

*윤청, **김대엽°
*충남대학교 컴퓨터 공학과
cyoun@cs.cnu.ac.kr, kdymn2@cnu.ac.kr

Workspace-based software configuration management system

*Cheong Youn, **Dae-yeob Kim°
*Department of Computer Engineering, Chungnam National University

요 약

본 논문에서는 워크스페이스 기반 소프트웨어 형상관리(Software Configuration Management, SCM) 시스템의 설계와 그 구현 방법에 대하여 논한다. 워크스페이스는 팀 단위의 협업 개발을 지원하기 위한 통합 환경으로, 형상항목(configuration item)에 대한 버전관리와 병렬 개발 프로세스를 제공한다. 본 연구는 이러한 워크스페이스 기반의 협업 개발 환경과 함께 프로젝트 관리, 변경 프로세스 관리 기능을 결합한 통합 소프트웨어 형상관리 시스템 구축을 목표로 한다.

1. 서 론

e-비즈니스 시대에 접어들면서 소프트웨어 개발 생명주기가 점점 짧아지고 있다. 또한 소프트웨어 활용분야가 산업 전 분야로 확산되면서 소프트웨어 품질의 중요성은 더욱 높아지고 있다. 이러한 추세에 따라 소프트웨어 개발과 관련된 거의 전 분야에 걸쳐 소프트웨어의 품질을 향상시키고 인터넷을 이용한 상호 정보교류와 의사소통을 지원하는 소프트웨어 협업개발 지원도구의 수요가 증가하고 있다.

소프트웨어 개발 프로세스에서 사용자로부터의 요구사항 변화나 엔지니어들에 의한 기술적인 부분의 수정은 소프트웨어를 구성하는 다양한 형상항목(문서, 소스코드 등)들의 모습을 수 차례 변화시킨다. 여러 가지 변경요구에 따라 팀 단위의 협업 개발환경에서는 구성원들에 의한 동일 형상항목의 변경이 빈번하게 발생할 수 있다. 이에 여러 구성원들의 동일 형상항목에 대한 변경을 체계적으로 관리하고 제어하는 것은 소프트웨어의 신뢰성을 향상시키기 위한 중요한 요소로 작용한다[1].

형상항목의 변경을 체계적으로 관리하기 위해서는 버전 관리가 필요하다. 단일 문서와는 달리 소스코드의 경우는 여러 개의 파일들로 구성된다. 소스코드의 특성상 파일들은 원래의 디렉토리 구조에 그대로 저장되어야 한다. 워크스페이스는 이를 위해 특정 디렉토리를 소스코드 관리도구에서 관리하는 특수 디렉토리로 지정하고 그 내부에서의 파일 변경 사항을 서버에 반영하여 통합된 버전관리가 가능하도록 한다.

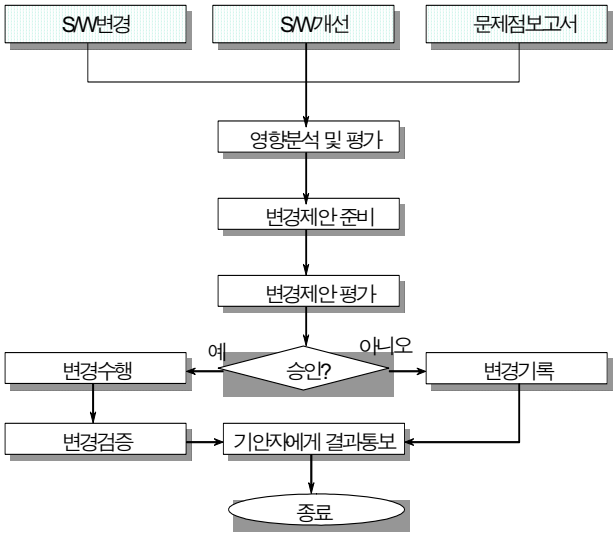
이에 본 논문에서는 팀 단위 협업개발 시 효율적인 형상항목 관리를 위한 워크스페이스 환경과 함께 프로젝트 관리, 변경 프로세스 관리 등을 포함한 통합 소프트웨어 형상관리 시스템을 제안한다.

2. 소프트웨어 형상관리

소프트웨어 형상관리는 소프트웨어 개발 및 유지보수 과정에서 산출되는 각종 결과물(문서, 프로그램, 하드웨어)들에 대한 계획, 개발, 운용 등을 종합하여 시스템의 모습을 만들고 이에 대한 변경을 체계적으로 관리, 제어하기 위한 활동이다. 유지보수 활동이 고객에게 소프트웨어가 인도된 이후의 변경 관리임에 비해 형상관리 활동은 소프트웨어 라이프사이클 전체, 즉 인도 이전의 변경 관리와 인도 이후의 변경 관리를 모두 포함한다[2].

IEEE에서는 “Software Configuration Management Plans” 표준을 정하고[3], SCM 시스템을 만들기 위해 계획 단계에서 필요한 사항들을 설명하고 있다. 또한 형상관리 활동을 형상식별(Configuration Identification), 형상 제어(Configuration Control), 형상상태 기록(Configuration Status Accounting), 형상 감사 및 검토(Configuration Audits and Reviews)와 같이 4가지로 나누고 있다. 형상 식별을 통해 소프트웨어 개발 과정에서 발생하는 모든 항목들은 그 기능적, 물리적 특성에 따라 문서화 되어야 한다. 또한 프로젝트 수준에서 제어해야 하는 소프트웨어 항목과 각 항목의 버전을 식별하는 규칙들은

이미 마련되어 있어야 한다. 소프트웨어를 개발해 나가는 과정에서 프로젝트 참여 인력의 작업 대상이 되는 형상항목들을 식별하는 작업은 앞으로 수행될 형상항목의 변경이나 감사 등에 대한 기초로 제공된다. 형상 제어는 형상 식별이 완료된 후, 형상항목들에 대한 변경을 평가, 협력, 승인/반려, 실행하는 업무들로 구성된 일련의 절차라고 할 수 있다. 소프트웨어의 각 형상항목들은 다수의 사용자들에 의해 공유되는 대상들이기에 이를 변경하기 위한 절차는 엄격한 규칙에 따라야 한다. 즉, 개인의 필요에 의해 임의로 특정 형상항목을 변경하는 것을 방지하기 위한 행정적 지침이라고 할 수 있다. 또한 형상항목에 대한 변경을 관리하기 위한 조직 내 구성원들의 역할 또한 미리 정의되어야 할 것이다. [그림 1]은 일반적인 형상항목 변경 절차를 나타내고 있다.



[그림 1] 일반적인 변경 절차

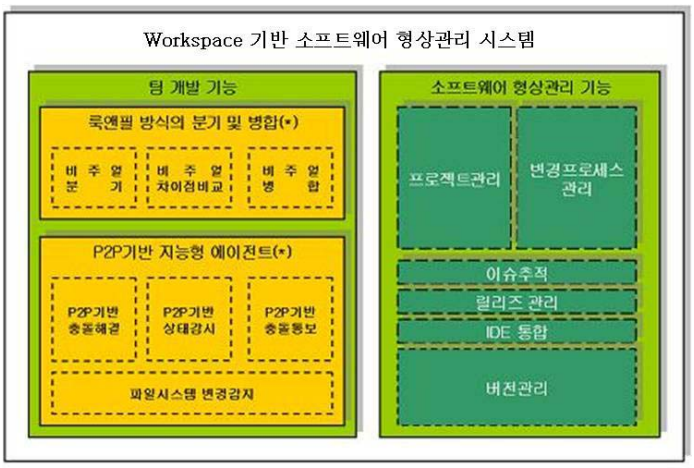
형상 제어의 핵심은 변경 요소를 식별하고 그에 대한 요청을 수행하는 것으로부터 시작한다. 변경 요청에 대한 평가가 이루어지고 그에 대한 승인/반려 여부에 따라 대상 형상항목은 변경이 실행되거나 그대로 유지된다. 변경의 실행은 형상관리 저장소로부터 해당 형상항목을 체크아웃(checkout)해야 이루어질 수 있다. 체크아웃이란 형상항목 변경자가 정당한 절차를 거쳐 저장소로부터 해당 형상항목을 획득하는 과정으로써 변경에 대한 권한을 획득하는 것이라 할 수 있다. 변경이 완료된 형상항목은 새로운 버전으로 새로운 기준선(Baseline)의 구성요소가 된다. 형상 상태 기록은 형상항목을 효과적으로 관리하는데 필요한 정보를 기록 및 보고하는 작업이다. 형상항목의 상태란 예를 들어, 승인된 형상 문서, 변경 상태, 변경에 대한 구현 정도 등의 내용이 될 수 있다. 이는 한 프로젝트에 속한 모든 구성원들에게 형상항목에 대한 이해를 제공하기 위한 수단이다. 형상 감사란 시스템의 최초 요구사항, 사용자의 별도 변경 요청사항이 최종 버전에 올바르게 반영되었는지를 확인하고 검토하는 과정이다. 또한 제품 패키지에 약속대로 수행되고 만들어진 컴포넌트들이 모두 잘 갖추어져 있는지 검증하는 단계이기도 하다. 형상 감사는 크게 두 가지로 구분하여 설명할 수 있다. 하나는 기능적 형상 감사(Functional Configuration Audit, FCA)로 요구사항 명세서에 정의된 기능들이 올바른 방법으로 모두 구현되었는지를 검사하는 작업이며, 다른 하나는 물리적 형상 감사(Physical Configuration Audit, PCA)로 형상항목의 일부로서 식별되는 모든 항목들이 제품의 베이스라인에 포함되었는지를 검사하는 것이다. 형상 감사는 일반적으로 개발 라이프 사이클의 가장 마지막 단계에서 수행되며, 이 과정을 통해 최종적으로 적합성을 인증 받는 것이 고객에게 전달된다. 또한 형상 감사는 품질보증(Quality Assurance) 부서나 고객 혹은 팀의 대표자들에 의해 이루어진다[4].

3. 워크스페이스 기반 소프트웨어 형상관리 시스템

본 연구에서 설계된 소프트웨어 형상관리 시스템은 프로젝트 관리를 중심으로 구성되어 있다. 프로젝트 관리는 소프트웨어 개발 기법에서 요구하는 프로세스를 중심으로 각 단계별 활동 및 형상항목의 관리를 포함한다. 프로젝트 중심의 형상관리는 현재 기업의 업무 프로세스, 각 단계별 기능, 그리고 조직 체계의 성능 및 기업 업무의 특성이 반영되어 기업의 조직, 프로젝트 및 프로젝트와 관련된 형상항목이 통합적으로 관리, 운영될 수 있도록 지원한다. 프로젝트 중심의 형상관리는 기업의 특성과 조직 형태, 다양한 프로세스 등을 고려하여 세분화 될 필요가 있으며 각 단계별로 산출되는 형상항목들과의 연관성을 나타낼 필요가 있다.

형상관리 시스템은 프로젝트 전체의 개발 과정에서 형상항목을 체계적으로 관리하는 것을 목적으로 한다. 프로젝트와 형상항목을 효율적으로 관리하기 위해 형상항목의 버전관리가 필수적으로 요구된다. 버전의 생성은 프로젝트 및 형상항목에 대한 단계적인 개발 상황과 기능적 향상을 가시적으로 확인할 수 있도록 도와주는 하나의 도구로서의 역할을 한다. 그러나 일반적으로 사용자에 의해 작성된 각각의 형상항목은 각 프로젝트에 포함된 여러 개의 태스크 안에 각기 존재하게 된다. 따라서 각각의 형상항목에 대한 변경은 그 내용과 시기가 서로 다를 수 있으므로 서로 다른 형상항목 버전으로 생성되어야 하고 변경 전과 변경 후의 형상항목은 서로 다른 수정 버전으로 구분된다.

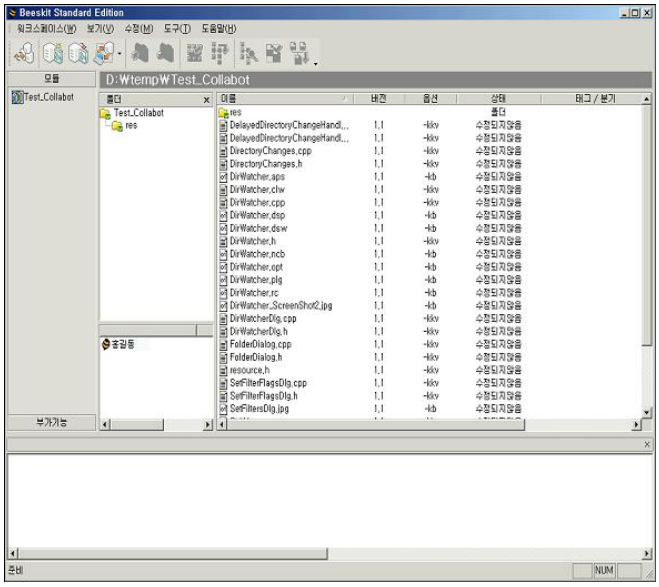
본 연구에서는 프로젝트 관리와 함께 형상항목의 효율적인 버전관리를 위해 [그림 2]와 같은 시스템 구조 제안한다.



[그림 2] 전체 시스템 구성도

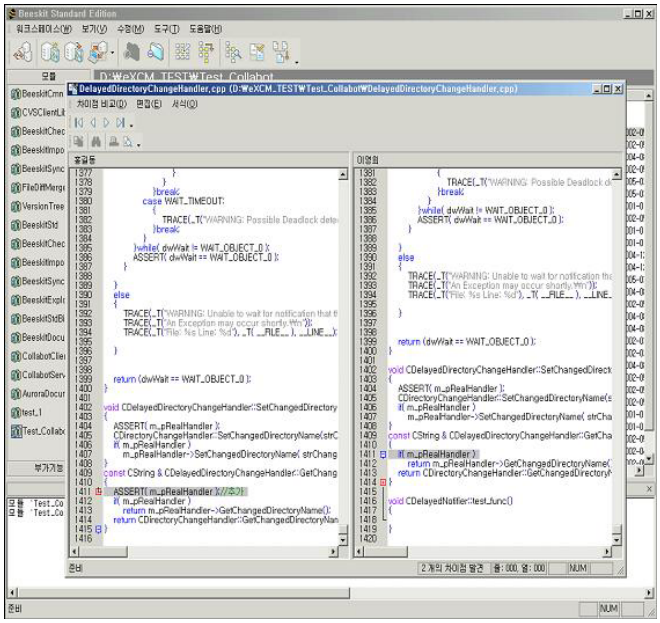
3.1 팀 개발 기능

소프트웨어 개발주기가 짧아지고 소프트웨어의 규모가 점차 확대되어 감에 따라 효율적인 협업 개발은 중요성은 날로 증대되고 있다. 더욱이 분산 개발환경의 확산은 지리적으로 떨어져있는 조직간 혹은 개발자 간의 유기적인 의사소통을 필요로 하고 있다. 팀 개발 기능은 워크스페이스 기반으로 관리되는 형상항목들에 대해 다수의 사용자가 접근하여 작업할 수 있는 통합 협업 개발 환경을 제공한다. 통합 워크스페이스 환경은 형상항목의 대상이 되는 자원에 대하여 구조적 형태와 변경 과정을 직관적으로 인식할 수 있도록 한다.



[그림 3] 윈도우 탐색기 스타일의 통합 워크스페이스

P2P기반 지능형 에이전트는 형상항목이 아직 공식화되지 않은 상태에서의 병렬개발 환경을 지원하기 위한 목적으로 개발되었다. 이는 동일 형상항목에 대해 동시 작업을 수행하고자 하는 사용자의 목록을 보여주고 P2P 기반으로 각자의 작업결과를 실시간 확인할 수 있도록 지원하는 역할을 수행한다. 공식화된 형상항목에 대해서는 전통적으로 Pessimistic 기법을 적용하여 동시 접근을 제한하지만, 공식화되지 않은 상태의 형상항목에 대해서는 Optimistic 기법을 적용하여 여러 사용자가 접근할 수 있도록 하였다[5]. 다만 동시접근으로 인한 충돌문제를 해결하기 위해 각자 수정한 내용에 대하여 상태감시를 하고 충돌에 대한 사건을 통보하도록 한다. 충돌이 일어난 형상항목에 대해서는 차이점 비교와 병합 기능을 제공함으로써 향후 서버에 업로드 되는 시점에서 통합된 결과를 반영할 수 있게 된다. [그림 4]는 충돌이 발생한 형상항목에 대한 차이점 비교 화면을 보여주고 있다.



[그림 4] P2P 공유 사용자와의 차이점 비교

일반적인 Optimistic 기법에서는 동일 자원에 대한 충돌의 여부를 서버에 업로드 하는 시점에 인식할 수 있는 반면 본 연구의 P2P 기반 지능형 에이전트는 충돌 여부를 미리 감지하여 해결하도록 함으로써 병합에 드는 시간적, 인적 자원의 낭비를 줄일 수 있다.

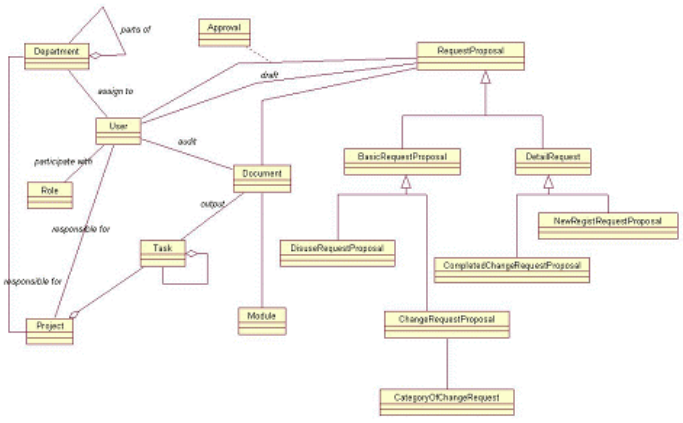
3.2 소프트웨어 형상관리 기능

팀 개발 기능에서는 공식화되지 않은 형상항목에 대해 다수의 사용자가 접근하여 작업할 수 있는 협업개발 환경에 대해 언급하였다. 소프트웨어 형상관리 기능은 프로젝트 관리와 변경 프로세스 관리, 버전 관리 등의 기능을 포함한다. 프로젝트 및 형상항목의 작성에 있어서 사용자들 사이에 발생할 수 있는 개발 내용 및 개발 과정에서의 의견 불일치와 병렬식 개발에 의한 중복수정 등과 같은 문제점을 해결하기 위해 잠금기능(lock) 및 잠금 해제 기능(unlock)을 제공한다. 이는 전통적인 동시성 제어(Cocurrent access control) 기법[6-7] 중 Pessimistic 접근법에 해당하는 것으로, 지정 체크아웃(reserved checkout)을 이용한 배타적 잠금(exclusive locking) 기법에 기반한다. 따라서 이미 공식화된 형상항목에 대해서는 변경 허가를 받은 후 직접 파일서버에서 Check-out을 한다. 특정 사용자에 의해 Check-out된 형상항목에 대해서는 잠금

기능을 제공하여 다른 사용자에 대한 수정과 변경을 불허한다. 그리고 사용자가 Check-out한 형상항목을 수정하여 다시 저장서버에 Check-in하는 과정에서는 형상항목에 대한 잠금 해제 기능을 실행하여 다른 사용자들이 필요에 따라 수정을 할 수 있도록 한다.

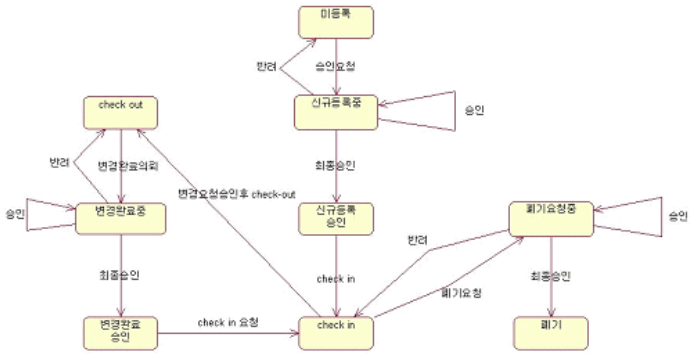
형상항목에 포함되는 파일을 관리하는 방식으로 단순 파일 첨부 방식과 모듈 연결 방식을 제공한다. 단순 파일 첨부 방식을 통한 파일 관리는 업로드 한 복수의 파일에 대한 개별적인 버전 관리가 어렵다는 단점이 있다. 반면 모듈 연결을 통해 서버에 업로드 및 관리하게 되면 파일 각각에 대한 버전 추적이 가능해 진다. 모듈 연결 방식은 클라이언트/서버의 변경 사항을 서로 동기화 시켜주기 위하여 클라이언트 측에 생성되는 폴더인 워크스페이스를 이용한다. 워크스페이스로 지정되는 폴더 아래에는 별도의 시스템 폴더와 그 아래 관리 목적으로 작성된 여러 파일이 생성되며, 이 시스템 폴더를 통해 서버 클라이언트 간 동기화가 가능해진다. 이러한 방식을 통해 사용자에 의해 수정된 워크스페이스 내 파일들은 새로운 버전을 갖게 된다.

2절에서 언급한 바와 같이 형상항목의 변경은 정당한 절차를 거쳐 수행되어야 한다. 즉 형상항목의 신규 등록이나 수정 등의 작업은 그것을 수행하기 위한 승인절차를 밟아야 가능하므로 각각의 경우에 해당하는 요청서를 작성해야 한다. 본 시스템에서는 요청의 형태를 크게 상세 요청과 기본 요청으로 구분한다. 상세 요청은 신규등록 요청과 변경완료 요청으로 구성되며, 기본 요청은 변경 요청과 폐기 요청으로 구성된다.



[그림 5] 클래스 다이어그램

프로젝트는 소프트웨어 제품의 생산에 요구되는 일련의 단계로 이루어진 프로세스의 집합이다. 각 단계마다 수행되어야 할 태스크와 그 단계에서 나와야 할 산출물인 형상항목이 정의된다. [그림 5]의 클래스 모델에서는 형상항목을 Document로 표현하였으며, 태스크는 경우에 따라 다시 하부 태스크를 포함할 수 있다. 그 외에 앞서 설명한 요청서가 형태에 따라 구분되어 나타나고 있다. 형상항목(Document)은 프로젝트를 진행하면서 얻어지는 작업 결과의 기본 단위이다. 따라서 형상항목이 처음 작성된 시점부터 폐기에 이르기까지 그 상태의 변화를 [그림 6]에서 표현하고 있다.



[그림 6] 형상항목 상태 변화도

5. 결론

소프트웨어 형상관리는 소프트웨어의 품질 향상을 위한 필수적인 요소이다. 대규모의 SI 업체나 솔루션 제공 회사들이 성공적인 프로젝트를 위해 ISO9001, SPICE, CMM 등과 같은 국제 품질 기준에 맞추어 프로세스를 진행하고 있는 추세이며, 그에 따른 소프트웨어 형상관리의 필요성이 날로 증가하고 있다.

본 연구의 형상관리 시스템은 지식정보 인프라를 구축하고 있는 인터넷을 통한 Web 환경을 지원하고 있다. 일반적인 기업의 형태가 인터넷에 의한 모든 지식정보의 공유라는 점에서 볼 때 Web을 통한 형상관리, 변경관리 및 문제의 추적을 지원할 수 있도록 하는 것이 가장 기본적이고 필수적인 요소라 볼 수 있다. 이러한 Web 환경의 지원은 사용자의 작업 환경에 대하여 총체적이고 글로벌적인 작업 공간을 제공하여 업무의 효율성을 극대화 시킬 수 있다. 또한 작업에 있어서 실 시간적인 요소를 충분히 반영하여 사용자들의 불필요한 시간적 비용을 줄임으로써 전체적인 작업 시간의 단축을 기대할 수 있다.

본 연구를 통해 프로젝트 관리, 프로세스 관리, 변경 관리를 지원함과 동시에 워크스페이스 기반의 버전 관리를 제공함으로써 통합 소프트웨어 형상관리 환경을 제공할 수 있게 되었다.

참 고 문 헌

[1] 김희철, “소프트웨어 생명주기에서 신뢰성 확보방안 연구”, 8 월, 2001

[2] 윤청, “소프트웨어 공학”, 생능 출판사, 1999

[3] IEEE, Standards Board, “IEEE Standard for Software Configuration Management Plans”, IEEE Std 828-1990

[4] Alexis Leon, “A Guide to Software Configuration

Management", Artech House, 2000
[5] 유재홍, 성미영, "공동저작 시스템에서의 동시성 제어와 쓰기권한 제어", 한국정보처리학회논문지, 제 7 권, 제 2 호, pp.347-354, 2000
[6] D. A. Menasce and T. Nakanishi, "Optimistic versus Pessimistic Concurrency Control Mechanism in Database Management Systems", Inform Syst, vol. 7, no. 1, pp.13-27, 1982
[7] Ciaran O' Reilly, Philip Morrow, David Bustard, "Improving Conflict Detection in Optimistic Concurrency Control Model", SCM-11 on ICSE 2003, 2003.5