소프트웨어 개발 프로젝트 제어를 위한 재작업 지표의 적용

한 혁 수[†] · 김 한 샘[††]

요 약

소프트웨어 개발 프로젝트는 성공률이 30% 밖에 되지 않는 어려운 과제이다. 소프트웨어 개발 프로젝트가 실패하는 이유는 여러 가지가 있을 수 있으나, 체계적인 관리 소홀이 큰 비중을 차지하고 있다. 특히, 완성도가 떨어지는 산출물을 다음 단계로 진행시키는 것은 많은 시간과 노력을 허비하여 프로젝트를 실패로 이끌 수 있다. 이를 방지하기 위해 채택되고 있는 방식은 동료 검토(Peer Review) 또는 인스펙션(Inspection) 등과 같은 산출물들에 대한 검토활동이다. 문제가 발견된 산출물들은 다시 개발자에게 돌아가서 수정하게 되는데, 이 과정을 재작업(Rework)이라고 한다.

프로젝트 관리자가 완성도가 떨어지는 산출물들을 다음 단계로 넘겨서 오류에 대한 막대한 비용을 지출하고 기간을 지연시키는 등의 사고를 막기 위하여, 본 연구에서는 재작업의 충실도를 높일 수 있는 방법을 연구하였다. 즉 프로젝트의 재작업 시에 작업분석을 시행함으로써 재작업된 결과의 검토 수준을 달리하는 재작업지표를 개발하였고, 이에 대한 검증을 위해 4개의 프로젝트를 선택하여 개발된 지표의 적용 여부를 관찰하고 그 효율성을 입증하였다.

키워드 : 소프트웨어 프로젝트 관리, 재작업, 인스펙션, 동료검토, 소프트웨어 측정, GQM

Applying rework indicator to control software development project

Hyuk-Soo Han[†] · Han-Saem Kim[††]

ABSTRACT

It is reported that the success ratio of software development projects has been only 30%. Many causes lower project's chance of success, particularly lack of systematic project management. Especially, moving on the next phase of project with unsatisfactory outputs can be very problematic because it can cause much waste of resource, time and even lead to the failure of the whole project.

Peer review and inspection are some of the practices designed to prevent such waste and possible failure. When defects are identified through such progress, each developer has to work on the product component again and fix the problem. This process is called rework.

In this paper, we propose a method for improving quality of reworked product component to prevent excessive cost and time consumed caused by moving on the next phase of a project with a problematic product component. More specifically, this paper suggests a rework indicator that measures the level of rework based on its complexity and severity and is used to choose appropriate checking method on reworked product component. The research also confirmed the method's usefulness and effectiveness by applying the suggested method on four projects.

Key Words : Software Project Management, Rework, Inspection, Peer Review, Software Measurement, GQM

1. 서 론

소프트웨어 프로그램의 크기가 작아서, 혼자서 또는 소수의 프로그래머가 프로그램을 개발해도 문제가 되지 않던 시절에는 프로젝트 관리가 그다지 큰 이슈가 아니었다. 하지만 요즘은 비행기 F-22의 소프트웨어 개발 비중이 80%에 이른다고 이야기 할 만큼 소프트웨어의 비중이 커지고, 참여하는 인력도 많게는 수백 명에 이르고 있다[1].

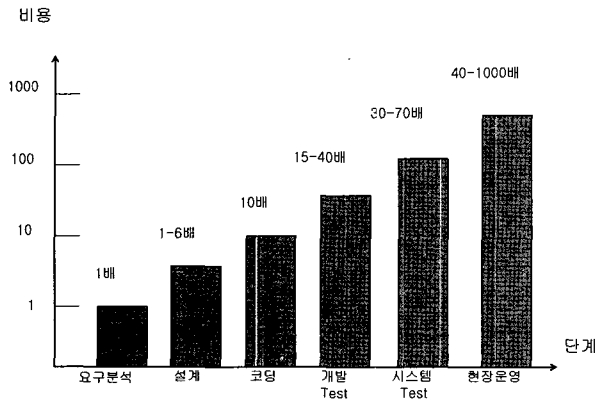
이렇게 소프트웨어의 프로젝트의 규모가 커지면서 소프트웨어 프로젝트는 매우 관리하기 어려운 과제가 되었다. 2003년에 발행된 미국의 IT프로젝트에 관한 조사 보고서(The Standish Group의 CHAOS Chronicles v3.0)에 따르면 2002년 미국에서 실시된 프로젝트 중 성공했다고 할 수 있는 IT프로젝트는 전체의 34%에 불과했다. 실패한 프로젝트의 51%는 품질, 납기, 비용 중 하나 이상의 항목을 만족시키지 못한 실패한 프로젝트이며, 15%는 중도에 멈춘 프로젝트라고 보고되어 있다[2]. 프로젝트의 성공률이 낮은 이유를 보면, 우수 프로그래머의 부족, 경험 없는 프로젝트 관리자, 요구사항의 잦은 변경, 산정의 어려움 등 일일이 열거하기가 어렵다.

이러한 어려움을 극복하기 위해 소프트웨어 공학 기법들이 연구되어 실행되고 있고, 프로젝트 관리 기법들도 만들어져서 보급되어 왔다. CMM/CMMI와 PMBOK같은 모델들

※본 연구는 상명대학교 자연과학연구소의 지원으로 수행되었음.
† 정 회 원 : 상명대학교 소프트웨어학부 교수
†† 준 회 원 : 상명대학교 대학원 컴퓨터과학과 박사과정
논문접수 : 2005년 12월 5일, 심사완료 : 2005년 12월 29일

도 채택되어 활용되어왔고, OOPD(Object Oriented Program Development), CBD(Component Based Development), PLD (Product Line Development)등의 개발 방법론들도 개발되어 사용되고 있다.

소프트웨어 개발은 생명주기를 기반으로 계획이 세워지고, 프로젝트 관리자는 주간 회의나 마일스톤 회의를 통해서 실제 작업이 계획대로 이루어지고 있는 가를 평가한다. 그러나 이러한 생명주기 모형과 다양한 관리기법을 사용하여 프로젝트를 계획대로 잘 진행하였더라도, 프로젝트의 막바지가 되면 초기에 발견하지 못하거나 수정하지 못한 결함들을 고치기 위해 대부분의 시간을 허비한다. 통합과정 후에도 발견되지 않은 결함들은 최종 제품이 나온 후에 문제점들이 발견 되어 더욱 어려운 상황에 빠지게 된다. 이러한 어려움은 (그림 1)에서 잘 보여주고 있다. 이러한 문제를 극복하기 위해 채택되는 방법이 동료검토, 인스펙션과 같은 산출물에 대한 검토활동이다. 이런 활동들을 통해 많은 문제들이 해결되고 있다.



(그림 1) 개발 단계별 오류 수정비용[3]

테스트나 검토 과정을 통해서 발견된 결함들은 다시 책임자에게 되돌려져서 수정되는데 이 과정을 재작업(Rework)이라 한다. 얼마나 많은 재작업이 수행되는 가는 프로젝트의 성공 여부를 결정할 수 있는 매우 중요한 요소가 될 수 있다. 따라서 재작업을 줄이고 수정된 부분에 대해서는 다시 결함이 발생되지 않도록 노력해야 한다.

프로젝트 관리 기법, 검토기법, 측정 등에 대한 다양한 연구가 있고, 프로젝트 초기의 오류를 줄여야 하는 등의 실증적 연구 결과가 나오고는 있지만 재작업 자체에 대한 구체적인 관리 활동에 대한 사례나 지침은 없는 것이 실정이다.

따라서 본 연구에서는 프로젝트 통제 및 제어를 위한 활동 및 메트릭에 대하여 연구하고 재작업 지표를 개발하였다. 재작업 지표는 프로젝트의 재작업 결정이 내려진 작업에 대하여 작업분석을 시행하여 재작업 결과에 대한 검토 수준을 달리하는 것이다.

발생된 오류에서 발전될 수 있는 추가적 오류 및 반복 오류를 줄이기 위하여 재작업 활동의 완성도를 높이는 것이 본 연구의 목표이며, 이는 결과적으로 프로젝트의 품질을 높이는 데에 많은 영향을 줄 것으로 기대된다.

본 연구는 다음과 같은 내용으로 구성된다. 2장 관련연구에서는 프로젝트 통제에 관한 연구와 검토 기법, 측정기법을 다루고 3장에서는 개발된 재작업 지표를, 4장에서는 검증을 위한 활동을 설명한다.

2. 관련 연구

2.1 프로젝트 통제 및 제어

프로젝트를 시작할 때 프로젝트 관리자가 처음 수행하는 작업은 프로젝트 계획을 세우고 이것을 문서화 하는 것이다. 프로젝트 계획서를 사실에 근거하여 제대로 작성하지 않으면, 단위 기간 별 목표의 설정, 자원의 배치, 기간의 설정 등 계획 내용이 맞지 않기 때문에 프로젝트의 실패확률이 커진다. 따라서 프로젝트 관리자는 과거의 프로젝트 수행의 경험과 측정결과들을 이용하여 수행할 프로젝트에 대하여 산정하고 이를 기반으로 계획을 세운다.

프로젝트가 수행되는 동안 프로젝트 관리의 범위는 일정, 비용, 품질, 인력, 외주, 변경, 문서, 보안, 위험 등 여러 가지 요소들이 될 수 있겠지만, 특히 일정, 비용, 품질에 대한 관리가 주 대상 범위라 할 수 있겠다.

일정이나 비용, 품질(오류)에 대한 관리를 위하여 프로젝트 관리자는 프로젝트의 내용을 계획과 요구사항에 맞추어 추적, 감독하게 된다. 이러한 활동을 프로젝트 모니터링이라 한다.

특히 프로젝트 모니터링 활동은 계획에 대한 적합성, 작업표준 즉 프로젝트에서 정해놓은 프로세스에 대한 준수성, 요구사항에 대한 준수성 등과 같은 내용이 검토되고 평가된다. 이러한 과정에서 계획에 대해 일정이나 비용에 대한 차이가 있을 경우 재계획을 통한 계획의 변경이 있을 수 있고, 프로세스에 대한 준수 및 요구사항에 대한 준수 등에 오류가 있을 경우에는 시정 요구가 발생하게 되며 재작업을 수행하게 된다. 이러한 프로젝트 관리에 대한 프로세스 표준은 CMMI, PMBOK 등에서 찾아볼 수 있다. CMMI에서 제시하고 있는 모니터링 활동의 영역인 PMC(Project Monitoring and Control)에서는 현재 진행되고 있는 소프트웨어 프로젝트의 진척도를 추적 및 감독하고 차후의 프로젝트 산정에서 사용하기 위한 데이터를 수집하기 위해 프로젝트 관리자가 따라야 할 프로세스와 절차가 설명되어 있다.

PMC의 주요 활동 항목은 측정 및 보고, 검토, 시정활동으로 구성되고 그 주요 대상은 다음과 같다.

- 프로젝트 작업산출물의 크기 또는 프로젝트 작업산출물의 변경
- 프로젝트의 공수와 비용
- 프로젝트의 주요 컴퓨터 자원
- 프로젝트의 일정
- 프로젝트 엔지니어링의 기술적인 활동
- 프로젝트의 비용, 자원, 일정, 기술적인 측면과 관련된 위험요소

직관적이고 합리적인 모니터링 활동을 위해 프로젝트 활동에 대한 측정 및 보고 활동이 수행된다. 이는 실제 진행 상황과 산정된 데이터와의 비교를 통해 프로젝트 가시성을 보장하고 프로젝트의 진행사항이 계획에서 벗어났을 때 효과적인 시정활동을 하기 위한 것이다.

계획된 일정에 따라 마일스톤 검토를 수행하여 공약변경 및 계획대비 편차가 큰 경우에 문서화된 절차로 시정조치를 수행하며 계획을 변경한다[4].

2.2 검토 활동

검토 활동이란 소프트웨어 품질 보증 활동의 한 부분으로 소프트웨어 개발 동안 발생하는 오류(error)와 결함(detect)을 조기에 발견하여 이들이 소프트웨어 개발의 다음 단계로 전파되는 것을 방지함으로써 소프트웨어의 품질을 높이는 것이다. 이러한 활동은 소프트웨어 개발 주기 어느 단계에서나 진행될 수 있다.

검토 활동은 검토의 방식이나 형식적인(formal) 정도에 따라 몇 가지로 구분되기도 하는데, 비형식적인 검토활동인 동료 검토와 보다 형식적인 검토 활동인 인스펙션이 그 대표적인 예이다.

2.2.1 동료 검토(Peer Review)

동료 검토는 일반적으로 형식 없이 검토가 필요할 때마다 동료들이 작업 산출물을 검토 하는 방법을 말한다. 정해진 검토 인원이나 제한 시간, 검토 리더는 없으며 동료와 의견을 나누듯이 검토를 진행한다. 동료 검토의 장점은 자유로운 의견을 통해 다양한 아이디어를 얻을 수 있고, 검토 과정을 통하여 지식이 동료에게 전파되는 것이다. 하지만 검토 리더가 정해져 있지 않아 정상적인 검토가 진행되지 않거나 일반 회의처럼 다수의 의견이나 힘있는 자의 의견에 전체의 흐름이 동조될 수 있다는 단점이 있다[4].

2.2.2 인스펙션(Inspection)

인스펙션은 동료 검토보다 형식을 갖춘 검토 방법으로써, 검토 인원, 준비 시간, 검토 시간, 검토 리더 등이 정해져 있다. 보통 세 명에서 다섯 명이 검토 인원으로 참가하며, 두 시간 이내에서 미리 검토 준비를 하고, 검토 회의 시간 역시 두 시간 이내로 한다.

검토 회의에는 검토 리더, 검토 인원과 산출물 제작자가 참여하게 되며, 검토 인원은 미리 준비한 문제점을 제기하며, 타당한 내용은 기록을 하게 된다. 검토 회의 마지막에는 검토의 대상이 된 산출물을 수용, 포기, 또는 수정 후 수용할 것인지에 대한 결정을 한다. 인스펙션 활동에 대해서는 지침이 확립되어 있어야 하고 담당자가 정해져서 이행해야 한다[5].

2.3 메트릭 개발의 기법-GQM

프로젝트 관리를 위해서는 현재의 진행되는 상황을 파악해야 하고, 이를 계획과 비교하여 그 차이를 기반으로 현재 프로젝트의 상태를 판단한다. 이때에 활동이나 산출물들의 상태를 알고 정확한 판단을 위해서 관련 데이터를 측정하고 메트릭을 사용한다. 메트릭은 일종의 함수로서 측정된 데이터를 이용하여 계산된다[6, 7]. 예를 들면, 교통 공학자들은 통행 차량 운행속도와 거리를 측정하여 특정 지점까지의 소요시간을 계산하고, 자동차 공학자들은 킬로미터 당 소요되는 기름의 양을 계산하는 연비라는 메트릭을 사용하여 차의 효율을 나타낸다.

메트릭을 개발하는 방법으로 많이 사용되는 것이 GQM (Goal-Question-Metrics) 방법이다. GQM은 측정 지표를 구하고자 하는 Business Goal을 선택하고, 그와 관련된 질문을 만들고 그 질문의 답이 되는 측정을 수행하는 방법으로써 다음과 같은 3단계로 이루어진다.

- 목표(Goal) : 개선 활동과 연관된 대상의 범위를 정의한다. 여기에는 제품, 프로세스, 자원 등이 포함된다.
- 질문(Question) : 목표 달성과 관련된 사항들을 특성화하기 위한 질문을 개발한다.
- 측정(Metric) : 질문과 관련된 측정치들을 나열하고 질문에 답하기 위한 연관된 메트릭을 개발한다[8, 9].

2.4 기존의 연구에서의 문제점

성공적인 프로젝트를 위해서는 앞서 얘기한 바와 같이 프로젝트의 관리활동 즉, 프로젝트에 대한 정확한 예측 및 계획, 계획에 맞는 활동을 하는가에 대한 모니터링 및 통제활동, 이러한 활동들에 기반이 되는 측정활동 등의 관리 프로세스들이 있다[10]. 이러한 프로세스와 관리활동들은 프로젝트에 유입되는 오류를 예방하고, 발생된 오류에 대해서는 초기에 발견하고 재작업하여 오류 수정의 비용을 줄이고자 함이다.

그러나 기존의 방식에서는 모든 재작업된 산출물에 대하여 획일적인 검토방식을 적용하여 재작업된 산출물의 중요도나 구현의 어려움이 반영되지 않았고, 새로운 오류의 유입이나 기존 오류의 잔류 등에 대한 체계적인 검토가 이뤄지지 않았다.

따라서 본 연구에서는 재작업 지표를 개발하여 재작업된 산출물을 분석하고 이에 대한 검토 수준을 달리하여 재작업 활동의 완성도를 높이는 프로세스를 개발하였다.

3. 재작업 완성도를 높이기 위한 재작업 지표의 개발

재작업 활동에 대한 완성도를 파악하는 측정치를 찾기 위해 본 연구에서는 GQM 방법을 이용하였다. GQM을 이용하여 재작업 지표를 개발하는 단계는 다음과 같이 총 네 단계로 구성된다.

3.1 GQM 방법을 적용한 측정치들의 개발

1 단계 : 비즈니스 목표(Business Goal)의 선택

비즈니스 목표 설정은 GQM 방법에서 가장 먼저 진행해야 할 단계로서 GQM 실제 목표 추출의 기초가 된다. GQM

활동을 수행하게 된 근본적인 목표를 생각하는 단계이다[9]. 본 연구에서 목표하는 바는 재작업 활동에 대한 시간이나 비용을 투자함으로써 완성도가 떨어지는 산출물들이 후반으로 넘어가는 것을 막고, 결함의 재유입으로 인한 부작용과 비용을 줄이기 위한 것이다. 그러므로 현 단계에서 비즈니스 목표는 다음과 같다.

- 재작업의 완성도를 높여 프로젝트의 성공률을 높인다.

2 단계 : 핵심성공요소(CSF) 선정

두 번째 단계는 첫 단계에서 설정한 비즈니스 목표의 핵심성공요소(CSF)를 선정하는 것이다. 이 핵심성공요소는 실제 GQM의 목표가 된다. 실제 프로젝트에서 고려할 수 있는 질문이나 측정치들을 식별할 수 있도록 보다 세부적으로 목표를 정의한다[11]. 이번 작업에서 선택된 목표는 다음과 같다.

- 재작업 시 새로운 오류의 유입과 기존 오류의 잔류를 검증한다.

3 단계 : 질문 및 측정

목표의 달성을 위해 요구되는 특성들을 알아내기 위한 질문과 측정치는 다음과 같다

- 재작업 전반
  - 작업이 수행되는 환경은 무엇인가?
    - 프로젝트 이름
    - 프로젝트 팀
    - 프로젝트 팀원의 수
    - 프로젝트 기간
  - 요청된 재작업은 어떤 특성을 갖고 있는가?
    - 재작업 내용(코드, 비코드)
    - 난이도(코드의 경우)
    - 결합도(코드의 경우)
    - 예상 작업기간(일/월)
    - 예상 수행 인원(명)
    - 재작업 횟수(회)
    - 해당 단계(전체 생명주기 중) 및 단계에 대한 내용(단계)
- 재작업의 검토
  - 재작업은 얼마나 완료되었는가?(작업의 모든 오류는 수정이 되었는가?)
    - 발견된 오류의 개수(개)
    - 수정된 오류의 개수(개)
    - 오류의 수정 여부(예/ 아니오)
    - 재작업은 어느 수준으로 검토되어야 하는가?
    - 재작업의 중요성
    - 재작업의 검토 수준

4 단계 : 지표의 결정

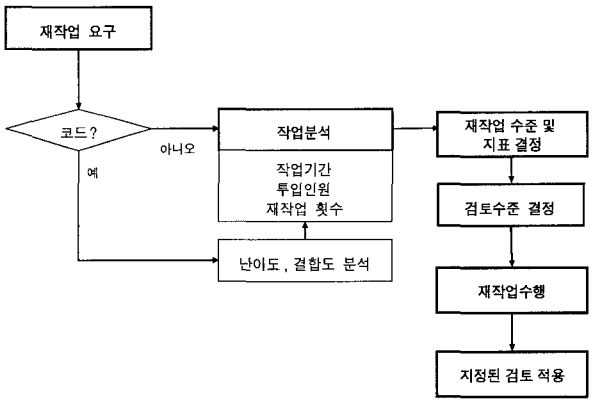
마지막으로 네 번째 단계는 위의 단계를 통하여 지표를 결정하는 것이다. GQM을 통한 재작업에 대한 정보의 체계화를 통해 재작업에 대해 식별되어야 할 내용을 파악하였고 재작업의 중요성을 결정하기 위한 지표가 요구된다는 것을 알 수 있었다. 지표에 대한 설명은 다음 장에서 자세히 다루어 진다.

3.2 재작업 지표

작업의 중요도와 코드의 중요도를 인지하여 검토를 철저하게 하는 것은 초기 진도율에는 역효과가 날 수 있지만, 오류의 잔류나 재유입을 방지함으로써 결과적인 진도율에 도움이 될 수 있다. 따라서 본 연구에서 개발한 지표는 재작업의 완성도를 높이기 위한 작업이므로 재작업 지표라 하였다. 재작업 지표의 활용 과정 및 재작업 완성도 지표에서 사용되는 요소들 각각에 대해 살펴보면 다음과 같다.

3.2.1 재작업 지표의 활용 프로세스

재작업에 대한 GQM을 실시한 결과 재작업 활동의 완성도를 높이기 위해 고려해야 하는 단계는 두 가지 부문으로 축약할 수 있었다. 한 가지는 재작업 해야 하는 대상에 대한 중요도를 결정하는 것이었고, 다른 하나는 이에 대해 검토의 수준을 달리하는 것이다. 작업의 중요도를 파악하기 위해 작업을 분석하고 분석된 결과에 따라 검토의 수준을 달리하는 과정을 포함하여 프로젝트를 수행하면 프로젝트의 완성률 및 진도율에도 향상이 있을 것으로 가정하였다.



(그림 2) 재작업 완성도 확인 프로세스

3.2.2 재작업 수준(reWork Level : WL) 결정

재작업 수준을 결정하기 위한 판단기준으로 GQM방법을 통해 나타난 재작업의 특성 중 난이도, 결합도, 재작업횟수, 투입인원, 작업기간 총 다섯 개의 결정 요인을 선택하였다.

<표 1>의 다섯 가지 요소는 동일한 비중으로 작업에 영향을 준다고 가정하여 재작업 수준에 대한 메트릭을 다음과 같이 작성하였다.

$WL=(D+C+P/T_p + D/T_d + n)/5$

(비코드 경우 : $WL= (P/T_p + D/T_d + n)/3$)

〈표 1〉 재작업수준 결정 요소

결정 요인 유형	설명	값(Value)					
난이도(Difficulty)	재작업을 위해 필요한 기술 정도. 재작업의 복잡한 정도-코드일 경우	하 0		중 0.5		상 1	
결합도(Coupling)	모듈 사이의 상호 연관성의 복잡도-코드일 경우	없음 0	데이터 0.6	제어 0.7	외부 0.8	공통 0.9	내용 1
투입인원(Number of Developers)	프로젝트 수행인원(Tp) 대비 작업 수행에 관련되는 예상 인원의 수(P).	P/Tp					
작업기간(Ratio of Rework Duration)	재작업이 속한 단계(phase)의 기간(Td) 대비 작업을 수행할 예상 기간(D)	D/Td					
재작업 횟수(Rework Count)	동일한 작업에 수행된 이전의 재작업 횟수	n					

3.2.3 재작업 지표(Rework Index)와 검토 수준(Review Level : RL)

지표란 상태가 가늠이 되는 표시 또는 특징이다[11]. 재작업의 수준에 대한 지표의 설정을 위해 재작업 측정치들과 검토수준간의 관계를 고려하였다. 작업의 오류 확률이 높고 프로젝트에 영향을 많이 미칠 수 있는 사항들은 다음과 같다.

- 재작업의 횟수
- 자원의 사용
- 재작업 산출물의 변화로 인한 다른 산출물의 변경 가능성

위의 사항들 중 특히, 재작업의 횟수는 오류의 재유입을 의미하며, 재작업의 완전성과 직접적인 관련이 있는 항목으로서 가장 심각하게 고려되어야 하는 요소이다. 따라서 최소한의 결합도 및 자원의 사용으로 요인을 통제한 경우 재작업의 횟수에 의해 WL이 달라지는 정도에 따라 검토 수준을 결정하고, 수준을 부여하여 지표를 설정하였다.

〈표 2〉 재작업 지표와 검토수준

WL	재작업 지표	검토수준
0 ~ 0.2 미만	1	담당자 완성도 여부 결정
0.2 ~ 0.5 미만	2	동료 검토
0.5 ~ 1, 1이상	3	인스펙션

4. 사례를 통한 검증

본 연구의 가설을 검증하기 위해 학부 4학년 프로젝트를 대상으로 재작업 지표를 적용하였다. 4개월 과정의 프로젝트 4개를 선택하여 2팀은 지표를 사용하였고, 다른 2팀은 재작업 지표를 사용하지 않은 채로 프로젝트를 진행하였다.

연구의 타당성 및 결과의 신뢰성을 저해할 수 있는 우연적 조건들을 통제하고 실험 대상들의 수준을 유사하게 맞추기 위해 설정한 프로젝트 진행의 개요는 <표 3>과 같다.

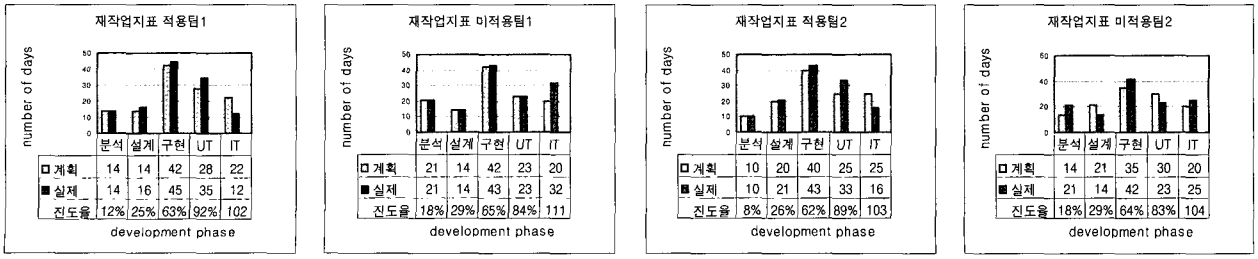
〈표 3〉 프로젝트 개요

개발 주제	쇼핑몰 개발 - 관리자와 고객 양측의 접근 모드 모두 포함 - 조건의 통제를 위해 요구사항서를 개발 전 프로젝트 팀에 제시함
생명주기	피드백을 포함한 폭포수 모형
개발과정	4개월
팀원	4~5명
개발 산출물	프로젝트 계획서, 요구사항 명세서, 설계명세서, 테스트 결과보고서, 제품

** 참여 인원의 수준을 맞추기 위해 학습수준, 프로젝트 경험, 사용 언어를 유사하게 한다.

샘플로 선정된 프로젝트에서 관찰해야 할 것은 두 가지였다. 프로젝트의 진도율 및 프로젝트의 완성률, 즉 품질이다. 본 실험에서 진도율을 확인하기 위해서는 계획대비 실제 일정에 대한 기록을 확인하였고 완성률을 확인하기 위해서는 최종 산출물의 요구사항에 대한 적합성(요구사항 수에 대한 최종 결과의 만족 개수)을 확인하였다. 완성률을 보기 위해 제안된 요구사항을 체크리스트로 만들어서 요구사항의 적용 여부를 확인하였다.

프로젝트 별 완성률은 <표 4>와 같다. 표에서 보는 바와 같이 재작업 지표의 적용 여부에 따라 완성률의 차이가 나는 것을 볼 수가 있었다.



(그림 3) 진도율의 확인

〈표 4〉 완성률의 확인

재작업지표 적용팀1	97%
재작업지표 적용팀2	94%
재작업지표 미적용팀1	95%
재작업지표 미적용팀2	88%

5. 결 론

프로젝트 관리의 어려움을 극복하기 위해 소프트웨어 공학 기법들과 프로젝트 관리 기법들이 개발되어 사용되어 왔다. 하지만 여전히 소프트웨어 제품의 보이지 않고 만져지지 않는 특성으로 인해, 개발된 결과물에 대한 품질에 대해 구체적인 가시성을 확보하고 그 결과를 확신하기가 어렵다. 따라서 소프트웨어 개발 활동 중 많은 부분을 재작업 활동이 차지하고 있는 것이 사실이다.

완성도가 떨어지는 산출물을 다음 단계로 진행시키는 것은 많은 시간과 노력을 허비하여 프로젝트를 실패로 이끌 수 있다. 특히 많은 프로젝트의 문제는 각 팀원들이 단위 모듈을 구현하고 단위 테스트가 끝난 후에 통합작업을 하면서 발생된다. 통합 시에 인터페이스 관련 에러를 해결하고 나서 논리적 에러가 발생되면, 다시 단위모듈을 체크아웃하여 그 동안 했던 작업을 다시 하면서 많은 시간과 노력을 소비하게 된다. 더 나쁜 상황은 통합과정에서 발견되지 않은 결함들이 최종 제품이 나온 후에 발견 되어 재작업을 하는 경우이다.

본 연구에서는 이러한 문제를 해결하기 위하여 프로젝트 통제 및 제어를 위한 활동 및 메트릭에 대해 연구하여 재작업 지표를 개발하였고, 개발된 지표를 통해 작업의 중요성을 판단하고 검토수준을 달리하는 프로세스를 개발하였다. 이를 프로젝트에 적용하여 연구 결과의 효율성을 검증하였다.

추후 진행되어야 할 연구는 이 지표의 타당성을 부여하기 위해 다양한 프로젝트의 결과를 지표에 반영하고 재작업 지표를 더욱 정교하게 하는 것이다.

참 고 문 헌

[1] Wolfhart Goethert, Matthew Fisher, "Measuring Acquisition Process", SEPG 2002, 2002.

[2] The Standish Group International Inc., "Latest Standish Group CHAOS Report Shows Project Success Rates Have Improved by 50%", http://www.standishgroup.com/press/, 2003.

[3] Watts S. Humphrey, "A Personal Commitment to Software Quality", The Software Engineering Institute Carnegie Mellon University, 1994.

[4] CMMI Product Team, "Capability Maturity Model® Integration (CMMISM), Version 1.1, CMMISM for Software Engineering (CMMI-SW, V1.1)", CMU/SEI-2002-TR-029, 2002.

[5] R.Pressman "Software Engineering : A practitioner's approach", Addison Wesley, 2004.

[6] M.B.Chrissis, M.Konrad and S Shrum, "CMMI Guidelines for process integration and product improvement", Addison-Wesley, 2003.

[7] John McGarry, et al, "Practical Software Measurement-Objective Information for Decision Makers", Addison-Wesley, 2001.

[8] Donald R. McAndrews, "Establishing a Software Measurement Process", Technical Report CMU/SEI-93-TR-016, Software Engineering Institute, 1993.

[9] Rini van Solingen, Egon Berghout, "The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development", McGraw-Hill, 1999.

[10] William A. Florac, Robert E. Park and Anita D. Carleton, "Practical Software Measurement : Measuring for Process Management and Information". CMU/SEI-97HB-003, Software Engineering Institute, 1997.

[11] Wolfhart Goethert and Will Hayes, "Experiences in Implementing Measurement Programs", Technical Note CMU/SEI-2001-TN-026, Software Engineering Measurement and Analysis Initiative, 2001.

[12] James A. Rezum, "Defining and Understanding Software Measurement Data", Software Engineering Institute

[13] John H. Baumert, Mark S. McWhinney, "Software Measures and the Capability Maturity Model", Technical Report CMU/SEI-92-TR-025, Software Engineering Institute, 1992.

한 혁 수

e-mail : hshan@smu.ac.kr

1985년 서울대학교 계산통계학과(학사)

1987년 서울대학교 대학원 계산통계학과 (이학석사)

1992년 Univ of South Florida 전산학과 (공학박사)

2000년~2003년 시스템통합기술연구원 원장

2003년 소프트웨어진흥원 소프트웨어공학 연구소장

2004년~2005년 상명대학교 소프트웨어대학 학장

1993년~현재 상명대학교 소프트웨어학부 교수

관심분야 : 소프트웨어 프로세스, 소프트웨어 품질, 소프트웨어 사용성 평가 등

김 한 샘

e-mail : puru@smu.ac.kr

1997년 상명대학교 경영학과(학사)

2000년 상명대학교 정보통신대학원 멀티미디어학과(이학석사)

2003년 상명대학교 컴퓨터과학과(박사수료)

2003년~2004년 시스템통합기술연구원 연구팀 팀장

2002년~현재 상명대학교 소프트웨어학부 강사

관심분야 : 프로세스개선, 측정, 6시그마, 소프트웨어품질 등