소프트웨어개발 일정관리와 품질관리의 통합 방안

A Suggestion for Merging Quality Management into Software Project Schedule Management

백 선 욱 (Seonuck Paek) 상명대학교 컴퓨터소프트웨어공학과
한 용 수 (YongSoo Han) OSP 연구소
홍 석 원 (Sugwon Hong) 명지대학교 컴퓨터소프트웨어학과

요 약

소프트웨어 규모가 대형화 됨에 따라 개발에 소요되는 시간과 인력도 대형화 되고 있으며, 또한 원하는 수준의 소프트웨어 품질을 얻기 위해 필요한 테스트 비용도 점점 더 증가하고 있다. 소프트웨어 프로젝트 개발 과정에서 품질 관리를 위해 다양한 결함 제거 기법들이 사용되고 있으나, 이러한 결함 제거 기법과 결함 제거 시간이 전체 일정에 미치는 영향은 아직까지 체계적으로 분석되지 못하고 있다.

본 논문에서는 일정한 소프트웨어 품질 수준을 달성하기 위해 소요되는 시간을 소프트웨어 개발 일정 관리에 반영한 새로운 일정관리 모델을 제안한다. 제안된 모델은 CMU의 PSP/TSP (Personal Software Process/Team Software Process)를 적용하는 개발 과정에서 수집된 결함 정보, 결함 제거 시간, 평균 결함 제거시간 및 단계별 결함 제거율을 사용하여 일정 지연 여부를 실시간으로 추적할 수 있도록 하고 있다. 이를 위해 본 논문에서는 소프트웨어 품질 달성에 필요한 작업량을 일정 관리 측정체계와 동일한 측정체계에서 사용할 수 있도록 하는 품질 지수(Quality Value)를 새로 제안한다.

본 연구의 결과는 일정과 품질을 분리하여 관리하는 기존의 일정 관리 방법을 보완하여 프로젝트 관리자를 비롯한 모든 관계자가 품질 관리의 중요성을 인식하고 품질 저하 문제를 사전에 예방하는데 활용될 수 있을 것으로 기대된다.

키워드: 품질 관리, 일정 관리, PSP, TSP, Earned Value, Planned Value, EVMS, 품질 지수 (Quality Value)

Ⅰ. 연구배경

Standish group의 2000년 보고서에 의하면 소프트웨어 개발 프로젝트에서 평균적으로 63%에 이르는 일정 지연이 발생하였으며, 23%의 프로젝트는 완료되지도 못한 것으로 보고되었다(SGI, 2001). 이러한 일정 지연 및 프로젝트 실패의 주요 원인은 소프트웨어의 결함과 관련되어 있는 것으로 알려져 있는데, Jones(2000)의 통계 자료에 의하면 소프트웨어 개발 프로젝트에서 결함 발견 및 제거 작업은 대형 프로젝트의 경우 전체 작업의 36%까지 이른다. 그러나 이러한 높은 결함 관련 작업 시간에도 불구하고 소프트웨어 개발 작업에서는 비용증가, 일정지

연이 빈번하게 발생하며, 때로는 기대 이하의 품질로 인해 프로젝트가 종종 실패하거나 취소되기도 하는 상황이다.

따라서 소프트웨어 품질 관리를 프로젝트 성공의 핵심 요소로 간주하여 소프트웨어 개발 과정에서 품질 개선을 위한 다양한 기법들이 연구되고 있다. NASA/GSFC의 SEL(Software Engineering Laboratory)에서는 Validation, Verification, Object Oriented Concept, Cleanroom Theory와 Goal-Question-Metrics(GQM) Model 등의 다양한 연구 결과를 발표하여 소프트웨어 품질 향상을 지원해오고 있다(Basili 등, 2002). 또한, Zage 등(2003)은 대형시스템에서 각 단계별로 효과적으로 결함을 제거하는 방법을 고찰하고, 완전히 제거되지 않은 결함이 다음 단계로 전이되는 경우 약 8배 정도 비용이 증가함을 보여주고 있다. 한편, Liblit 등(2003)은 결함가능성 있는 정보를 고객의 불만사항으로부터 효율적으로 찾기 위한 방법을 제시하고 있다. 이러한 다양한 품질 관리 기법 중에서도 특히 결함 관리는 소프트웨어 개발에서 재작업을 줄일 수 있도록 하는 중요한 요소이므로(Krishnan 등, 2000), 결함 제거를 위해 실제 개발 과정에서 다양한 기법들이 사용되고 있다. 예를 들어, Carnegie Mellon 대학의 SEI(Software Engineering Institute)에서는 PSP/TSP(Personal Software Process/Team Software Process)를 사용해서 개발할 경우에 단위 테스트 이전에 review나 inspection 등의 결함 제거 기법을 사용하여 결함을 제거하면 계획에 비해 평균 6% 정도의 일정 지연만을 보이고 프로젝트가 완료되었음을 보고하고 있다(Davis 등, 2003; Conn, 2004). 즉, 품질을 대표하는 결함을 적시에 제거함으로써 프로젝트 일정지연을 방지할 수 있으며, 또한 결함이 적은 소프트웨어가 프로젝트 성공률도 높다는 것을 Jones(2000)와 Davis 등(2003)의 결과에서도 확인할 수 있다.

한편, 소프트웨어 개발 과정은 일반 공산품의 생산과정과 달리 분석, 설계, 개발, 테스트 등의 전체 과정에서 개발자 및 개발 조직의 역할과 관리가 중요하다. 따라서, CMMI를 비롯하여 다양한 프로젝트 관리 기법 및 도구들이 제안되고 적용되고 있다(Curtis 등, 2001; Liu 등, 2003; Chrissis 등, 2003; SEI, 2004). Liu 등(2003)은 소프트웨어 품질 향상과 프로젝트 관리를 위한 사전 경고 시스템을 제안하였는데, 이 시스템에서는 Fuzzy Logic을 활용하여 품질, 일정, 비용에 관한 위험을 분석하고 위험을 0와 1사이에서 표시하고 있다. 또한, Jalote(2000), 장시영과 신동익(2000)에서는 기존의 제조업이나 건설업에서 사용되던 EVMS(Earned Value Management System)를 소프트웨어 개발 과정에 적용하여 비용과 일정 등을 통합 관리하는 방안의 타당성에 관한 연구도 제안되고 있다. 또한, Solomon (2002)는 CMMI을 적용한 프로세스 모델에서 프로젝트 관리의 효율성을 높이기 위해서 일정과 비용 관리에 EVMS의 적용 가능성을 검토하고 기술적인 측면을 통합한 EVMS 모델을 제시하고 있다. 그러나 이러한 프로젝트 관리 모델에서는 일정의 지연 여부, 비용의 과다 여부만을 관리하며, 일정한 소프트웨어 품질을 유지하기 위해 필요한 결함 제거 시간을 반영한 통합 일정 관리에 대해서는 논의되고 있지 않은 실정이다.

본 논문에서는 소프트웨어 프로젝트의 일정 관리에 품질 요소를 반영할 수 있는 새로운 품질-일정 관리 모델을 제안 한다. 본 논문의 모델은 일반적인 프로젝트 관리 기법에도 적용될 수 있지만, 제시된 모델에 필요한 자료 수집의 편의상 PSP에서 제공되는 일정 관리(EVT: Earned Value Tracking)(Watts,1995; ANSI, 1998)에 품질 요소를 함께 다룰 수 있도록 적용하는 방안을 제안하고자 한다. 이를 위해 본 논문에서는 결함 제거에 필요한 시간을 품질 지수(*QV*: Quality Value)라는 개념으로 도입하여 프로그램 개발 과정에서 최종 제품의 결함 발생 가능

성을 예측하고 이를 바탕으로 이러한 결함제거에 필요한 작업량이 프로젝트 일정에 어떠한 영향을 미칠지를 분석할 수 있도록 하고자 한다. 대형 소프트웨어의 개발에서 개발자들은 단위 프로그램을 개발하고 각 프로그램은 PSP의 작업기준에 따라 단위 테스트(unit test)를 하고 테스트과정에서 발생한 결함을 기록하도록 하며, 이러한 단위 프로그램의 테스트 결과를 수집하여 전체 프로그램의 품질과 일정지연을 예측하고 EVM(EV Management)에 적용하여 품질을 고려한 일정 관리가 될 수 있도록 하고자 한다.

본 연구에서는 PSP의 자료 기록 방법에 따라 데이터가 수집되며, 프로젝트 계획 수립 과정은 TSP에 따르며, 단위 프로그램 진행 결과는 수시로 취합된다고 가정한다(Humphrey, 1995; Humphrey, 2000). 일정 관리를 위한 작업량 예측과 일정 관리 기법은 PSP 1.1의 일정계획(Task & Schedule Planning)에 따르고 일정 추적 방법에 있어서 PSP1.1의 EVM을 적용한다.

본 논문의 Ⅱ장에서는 PSP/TSP의 품질 관리와 일정 관리에 대해 간략히 기술하고 Ⅲ장에서는 본 논문에서 제안한 통합 모델을 위한 품질 지수 산정 모델에 대해 기술한다. 또한, Ⅳ장에서는 품질 지수 모델을 프로젝트의 일정 예측에 활용할 수 있는 방안을 제시하고 Ⅴ장에서는 본 논문의 모델을 실제 개발 업무에 적용하기 위해 필요한 데이터 수집 방안을 PSP와 관련하여 기술한다.

Ⅱ. PSP/TSP 결함 관리와 일정 관리

2.1 결함 제거 효율과 결함 제거 시간

소프트웨어 공급 후의 소프트웨어 품질은 일반적으로 소프트웨어를 공급한 후의 일정 기간 동안 발견되는 결함의 양으로 표시할 수 있다(Goldenson 등, 2003). 여기서 결함의 양은 발견된 결함 수에 대해 프로그램 크기로 나누어 계산하고 defects/LOC[1] 또는 defects/FP[2]로 표시된다. 일반적으로 인수 테스트에서는 전체 결함의 약 40% 정도만 발견할 수 있는 것으로 알려져 있는데, 이 것은 Caper Jones가 다양한 프로젝트를 분석하여 얻은 결과이다(Jones, 2000). 이러한 결함 발견 가능성을 "결함 제거 효율(Defect Removal Efficiency)"이라 하며 다음과 같이 표시 한다(Jones, 2000; Humphrey, 1995):

$$\text{결함제거효율} = \frac{\text{제거된 결함의 수}}{\text{결함제거 전의 전체 결함의 수}} \times 100$$

일반적으로 소프트웨어의 결함 제거는 테스트에 의존하고 있는데, 테스트를 반복함에 따라 품질은 좋아진다. 그러나 이러한 테스트에만 의존한 결함 제거는 많은 작업량을 요구하고 프로젝트 일정 지연의 주요 원인이 된다는 것이 알려져 있다(Humphrey, 1997). 테스트에 의존한 결함 제거 보다 Review나 Inspection을 활용한 결함 제거 기법으로 보다 일찍 결함을 제거하는 것이 프로젝트 전체 작업량을 줄이는데 효과적이라고 알려져 있다(Humphrey, 1997). 이러한 결함 제거 방법들 간의 효율성을 비교하기 위한 방법으로 결함 하나를 제거하는데 필요한 작업량을 "결함 제거 시간"이라 하며 다음과 같이 나타낸다:

$$\text{결함제거시간} = \frac{\text{각 단계에서 결함제거에 사용된 작업시간}}{\text{발견된 결함의 수}}$$

Humphrey(1997)에 의하면 단위 테스트 이전 단계에서 결함 하나를 제거하는데 소요되는 결함제거시간을 1이라고 할 때 단위 테스트 이후

1) LOC: Line of Code, 프로그램 소스 코드의 라인 수로 PSP에서 주로 이용된다(Humphrey, 1995).
2) FP: Function Point, 사용자 입장에서 프로그램의 크기를 산정하고 기술적.환경적 측면을 고려한 크기산정방식(Jones, 2000; IFPUG, 2003).

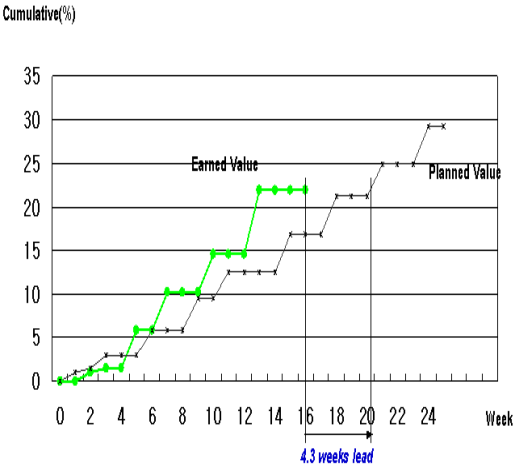
의 결함 제거 시간은 10이 되고, 고객이 사용 중에 발견한 결함을 수정하는 결함 제거 시간은 100에 달한다. 소프트웨어 결함은 평균적으로 100 LOC 마다 10 내외의 결함이 만들어 지고, 단위 테스트, 통합 테스트, 시스템 테스트와 인수 테스트에 의존한 일반적인 결함 제거 절차에 의하면 전체 결함의 83% 정도 제거되고 있다(Jones, 2000). 프로젝트에 적용되는 소프트웨어 개발 절차는 소프트웨어 품질(결함/크기)를 결정하는 핵심 요소다. 소프트웨어 프로젝트 관리에서 어떠한 개발 절차를 사용하여 짧은 작업 시간으로 결함을 최소화 할 것인가를 결정하는 것은 매우 중요하기 때문에 Review나 Inspection을 사용하여 단위 테스트 이전 단계에서 결함을 제거하는 것을 제안하고 있다(Humphrey, 1997; Jones, 2000).

본 논문에서 제안한 새로운 일정 관리 모델에서는 개발자에 관계없이 각 단계에서의 결함 제거 효율과 결함 제거 시간이 일정한 값을 갖는다고 가정하였는데, 각 개발자의 개발 능력에 대한 자료들이 수집되면 그 자료들을 바로 사용할 수 있도록 하였다.

2.2 PSP의 일정 관리

PSP의 일정 계획(PSP1.1)은 큰 프로그램을 개발하기에 앞서 그 크기를 예측하고 작은 프로그램으로 세분화하여 일정 계획을 수립하고 일정을 관리할 수 있도록 하는 방법을 제공하고 있다. PSP의 일정관리에서는 작업 계획량을 *PV* (Planed Value)로, 작업 완료량을 *EV*(Earned Value)로 표시하여 전체 작업에서 진행된 작업량을 표시하는 척도로 이용하는데, 일정과 함께 그래프로 표시하여 일정에 따라 누적 *PV*에 대해 누적 *EV*의 변화를 추적할 수 있도록 하고 있다(<그림 2.1> 참조). 이 때 사용되는 *PV*는 전체 프로그램을 개발하는데 필요한 작업량에 대해 세부작업 항목의 작업량을 백분율로 표시하는데, 세부작업이 완료되면 그 항목에 할당되었던 *PV* 값은 누적 *EV*에 합산된다. 프로젝트 시작 단계에서는 누적 *EV*는 0으로 표시되지만 모든 작업이 완료되면 누적 *EV*는 100이 된다. PSP에서는 해당 작업 항목이 완료 되었을 때만 그 항목에 할당된 *PV*를 *EV*에 적용되고 작업이 진행 중인 경우는 누적 *EV*에 합산되지 않는다.

TSP launch(Humphrey, 2000; Humphrey, 2002)과정에서는 프로젝트 팀 전체가 소프트웨어 개발에 필요한 작업량을 산정하고 업무가 한 사람에게 집중되지 않도록 조정한다. 계획된 일정은 각 참여자의 목표가 되며 매주 일정준수 여부는 팀에 보고되고 수집된 정보는 *EV*의 산정근거가 되어 *PV*와 *EV*를 일정에 따라 기록하는데 사용된다. PSP/TSP 일정 추적의 예가 <그림 2.1>에 나타나 있는데, 이러한 그래프를 이용함으로써 일정 준수 및 지연 여부를 파악할 수 있다.



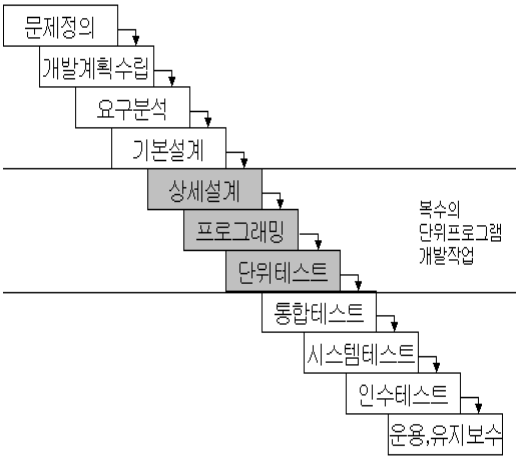
〈그림 2.1〉 *EV*와 일정

특정 시점에서 누적 *EV* 값이 누적 *PV* 값보다 더 크면 계획보다 일찍 완료될 가능성을 지표로 보여주지만, 그렇지 않은 경우 계획보다 더 지연될 가능성을 보여준다. <그림 2.1>에서는 예정보다 4.3 주 빨리 작업이 진행되고 있는 것으

로 파악되고 있는 예를 보여주고 있다.

Ⅲ. 품질 지수(QV: Quality Value) 모델

Ⅱ장에서 살펴보았듯이 품질과 일정은 밀접한 관계를 갖고 있는데, 이러한 상호 관계를 함께 표시하는 방법으로 본 논문에서는 품질 지수(*QV*)를 제안하고자 한다. *QV*는 품질이 일정에 미치는 영향을 표시하기 위한 지표로써, 단위 테스트 단계에서의 결함의 수를 활용하여 아직 제거되지 못한 결함의 제거에 필요한 예상 작업량을 예측하기 위해 사용된다.



〈그림 3.1〉 waterfall 모델

본 논문의 전개를 위해 먼저 소프트웨어 개발 과정은 <그림 3.1>와 같이 폭포수 모델을 따른다고 가정한다. 대형 소프트웨어 개발에서는 프로그램을 작은 단위로 분할하여 여러 개발자가 자신에게 분배된 단위 프로그램을 개발하고 점차 대형 소프트웨어로 통합하는 과정을 거친다. 이 과정에서 분석, 설계 과정은 통합적으로 수행하지만, 구현 즉 상세 설계, 프로그래밍 그리고 단위 테스트는 작게 분할된 프로그램 단위 기준으로 여러 개발자가 동시 다발적으로 수행한다. 이렇게 개발된 단위 프로그램을 모아 통합 테스트를 하고 시스템 테스트를 거친다. 그리고 프로젝트 마지막 단계로 인수 테스트를 한다. 개발된 소프트웨어는 고객에게 인도되어 운영 유지 보수 단계를 거친다. 본 논문에서 소프트웨어의 세부 기능에 대해 "상세 설계", "프로그래밍" 그리고 "단위 테스트"를 개발자가 반복적으로 하는 일련의 업무를 "단위 프로그램"이라고 한다.

기술의 편의상 프로젝트의 단위 테스트 및 그 이후의 단계를 다음과 같이 약어로 표기한다:

- *ut* : 단위 테스트(unit test)
- *it* : 통합 테스트(integration test)
- *st* : 시스템 테스트(system test)
- *at* : 인수 테스트(acceptance test)
- *ad* : 인수 후 운영유지보수(after delivery - 보통 6개월~12개월).

본문에 사용하는 기호를 다음과 같이 정의하는데, 아래 기호에서 첨자의 단계는 위에서 소개한 *ut, it, st, at, ad* 중의 하나를 의미한다:

$Df_{단계}$: 수정결함개수 - 각 단계에서 발견하여 수정한 결함의 수.

$Dt_{단계}$: 전체결함개수 - 각 단계 수행 전 이미 존재하고 있는 결함의 수로 각 단계에서 발견되어 수정된 결함과 미처 발견되지 못한 결함을 모두 포함. 이 결함은 해당 단계 이전에 생성된 결함만을 의미함.

$Di_{단계}$: 신규생성결함개수 - 각 단계에서 신규로 만들어지는 결함의 수.

$y_{단계}$: 결함제거효율 - 각 단계 시작 시점의 전체결함 중에서 그 단계에서 수정된 결함 수의 비율($Df_{단계}/Dt_{단계}$)(Ⅱ장 참조). 개발자 및 개발 팀에 대해 이 값은 일정한 값을 갖는다고 가정함.

$E_{단계}$: 결함제거작업량(시간) - 각 단계에서 결함

제거 작업을 수행하는데 필요한 작업량.

$e_{단계}$: 결함제거시간(시간/결함) - 각 단계에서 결함 하나를 제거하는데 소요되는 평균 작업량($E_{단계}/Df_{단계}$)(Ⅱ장 참조). 개발자 및 개발 팀에 대해 이 값은 일정한 값을 갖는다고 가정함.

$Ed(k)$: 단위 프로그램 k를 개발하는데 소요된 작업량(시간).

$PV(k)$: 프로젝트전체작업량 중에서 단위 프로그램 k에 할당된 개발 작업량의 비율.

$EV(k)$: 단위 프로그램 k의 완료 여부를 표시. 완료되면 $EV(k) = PV(k)$가 되고 아직 완료되지 않은 경우에는 $EV(k) = 0$.

먼저 단위 프로그램 k의 단위 테스트 단계에서 전체 결함의 수($Dt_{ut}(k)$)를 수정된 결함의 수($Df_{ut}(k)$)와 결함 제거 효율(y_{ut})로 표시할 수 있다:[3)]

$$Dt_{ut}(k) = Df_{ut}(k)/y_{ut}$$

단위 테스트를 마치고 난 후의 통합 테스트 시작 단계에서 단위 프로그램 k에 남아있는 결함의 수($Dt_{it}(k)$)는 단위 테스트에서 제거되지 못하고 남아있는 결함의 수와 단위 테스트 과정에서 신규로 생성된 신규 생성 결함의 수($Di_{ut}(k)$)로 표시할 수 있다:

$$Dt_{it}(k) = Dt_{ut}(k) * (1 - y_{ut}) + Di_{ut}(k)$$

이와 동일하게, 시스템 테스트, 인수 테스트와 운영 유지 보수 단계의 전체 결함의 수는 다음과 같이 표시된다:

$$Dt_{st}(k) = Dt_{it}(k) * (1 - y_{it}) + Di_{it}(k)$$

$$Dt_{at}(k) = Dt_{st}(k) * (1 - y_{st}) + Di_{st}(k)$$

$$Dt_{ad}(k) = Dt_{at}(k) * (1 - y_{at}) + Di_{at}(k)$$

운영 유지 보수 단계에서 단위 프로그램 k에 남아 있는 전체 결함의 수를 단위 테스트 단계에서 제거된 결함의 수와 각 테스트 단계에서 신규로 생성되는 결함 수의 함수로 표시하면 다음과 같이 표시된다:

$$\begin{aligned} Dt_{ad}(k) = {} & Df_{ut}(k)/y_{ut} * (1 - y_{ut}) * (1 - y_{it}) \\ & * (1 - y_{st}) * (1 - y_{at}) \\ & + Di_{ut}(k) * (1 - y_{it}) * (1 - y_{st}) \\ & * (1 - y_{at}) \\ & + Di_{it}(k) * (1 - y_{st}) * (1 - y_{at}) \\ & + Di_{st}(k) * (1 - y_{at}) \\ & + Di_{at}(k) \end{aligned}$$

단위 프로그램 k에 남아 있는 결함의 제거에 필요한 작업량($E_{단계}(k)$)을 전체 결함의 수($Dt_{단계}(k)$), 단계별 결함 제거 효율($y_{단계}$) 그리고 결함 제거 시간($e_{단계}$)으로 다음과 같이 표시할 수 있다:

$$E_{it}(k) = Dt_{it}(k) * y_{it} * e_{it}$$

$$E_{st}(k) = Dt_{st}(k) * y_{st} * e_{st}$$

$$E_{at}(k) = Dt_{at}(k) * y_{at} * e_{at}$$

$$E_{ad}(k) = Dt_{ad}(k) * y_{ad} * e_{ad}$$

이 값들은 결함을 제거할 때 각 단계에서 추가로 필요한 작업량을 표시하고 있으므로 본 논문에서는 "결함 제거 작업량"이라 정의한다. 각 단계에서 필요한 결함 제거 작업량(E)은 그 단계의 결함 제거 효율(y)과 결함 제거 시간(e)에 비례한다. 결함 제거 효율과 결함 제거 시간은 일반적으로 고정되어 있다고 가정할 수 있지만 각 단계에서 제거된 결함의 수는 전 단계의 결함 제거 결과에 의해 영향을 받는다(Jones, 2000).

위 식에서 각 단계에 신규 생성 결함이 없는

3) 본 논문에서는 모델 전개의 편의상 참여 개발자의 결함제거시간($e_{단계}$) 및 결함제거효율($y_{단계}$) 등이 서로 같다고 가정하였으나, 실제로는 달라도 관계없이 확장 적용 가능함.

경우에($Di_{ut} = Di_{it} = Di_{st} = Di_{at} = 0$), 각 단계의 결함 제거 작업량을 단위 테스트 단계의 수정 결함 수를 이용하여 표시하면 다음과 같이 정리할 수 있다:

$$E_{it}(k) = Df_{ut}(k)/y_{ut} * (1-y_{ut})^{*} y_{it} * e_{it}$$
$$E_{st}(k) = Df_{ut}(k)/y_{ut} * (1-y_{ut}) * (1-y_{it}) * y_{st} * e_{st}$$
$$E_{at}(k) = Df_{ut}(k)/y_{ut} * (1-y_{ut}) * (1-y_{it}) * (1-y_{st}) * y_{at} * e_{at}$$
$$E_{ad}(k) = Df_{ut}(k)/y_{ut} * (1-y_{ut}) * (1-y_{it}) * (1-y_{st}) * (1-y_{at}) * y_{ad} * e_{ad}$$

이 식은 단위 테스트 단계에서 수정된 결함의 수 $Df_{ut}(k)$가 많을수록 단위 테스트 후에 결함 제거와 관련된 작업량이 많다는 것을 보여주고 있는데, 이러한 결과는 Humphrey(1995)와 Jones(2000)의 자료와 일치한다.

결함 제거에 필요한 작업량(E)를 EV와 함께 사용할 수 있도록 하기 위해서 EV와 동일한 단위 체계로 정규화한다. 단위 프로그램 k에 계획되어 있는 작업량 $PV(k)$는 프로젝트 전체에 계획되어 있는 작업량 중에서 단위 프로그램 k에 계획된 작업량($Ed(k)$)을 백분율로 계산한 것으로 다음과 같이 표시 된다:

$$PV(k) = Ed(k)/(\text{프로젝트 계획 작업량}) * 100$$

통합 테스트 단계에서 단위 프로그램 k와 관련하여 필요한 결함 제거 작업량($E_{it}(k)$)을 $PV(k)$와 동일한 방법으로 정규화하여 $QV_{it}(k)$로 표시하면 다음과 같다:

$$QV_{it}(k) = E_{it}(k)/(\text{프로젝트 계획 작업량}) * 100$$

위의 두 식으로부터 통합 테스트에 대한 $QV_{it}(k)$는 다음과 같이 표시할 수 있다:

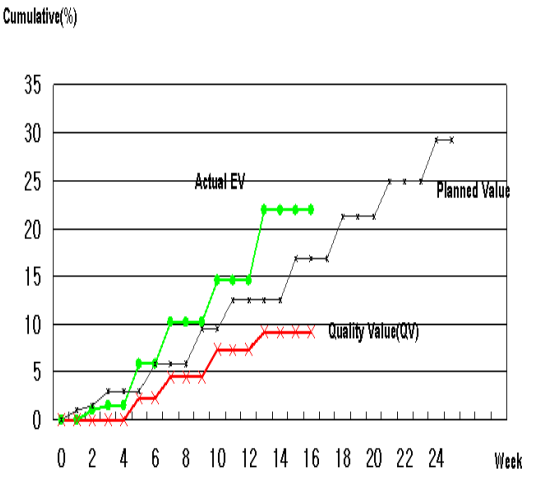
$$QV_{it}(k) = E_{it}(k) * PV(k)/Ed(k)$$

단위 프로그램 k가 완료되면 이 단위 프로그램에 계획되었던 $PV(k)$가 누적 EV에 합산되지만 이 단위 프로그램으로 인한 결함을 제거하기 위해서는 통합 테스트 단계에서 $QV_{it}(k)$에 해당하는 작업량이 추가로 필요함을 나타낸다. 시스템 테스트, 인수 테스트와 운영유지보수에 동일하게 적용하면 다음과 같다:

$$QV_{st}(k) = E_{st}(k) * PV(k)/Ed(k)$$
$$QV_{at}(k) = E_{at}(k) * PV(k)/Ed(k)$$
$$QV_{ad}(k) = E_{ad}(k) * PV(k)/Ed(k)$$

일정 추적(Earned Value Tracking)에서 QV를 이용하기 위해서는 개발이 완료된 단위 프로그램 각각의 QV를 누적하여 사용한다. 단위 프로그램 들 중에서 현재까지 개발이 완료된 것들의 집합을 C라 하자. 일반적으로 소프트웨어 개발 프로젝트는 인수 테스트 단계까지 진행되므로 이 과정까지의 QV 값을 누적하여 합산하면 다음과 같다:

$$\sum_{\forall k \in C} QV(k) = \sum_{\forall k \in C} QV_{it}(k) + \sum_{\forall k \in C} QV_{st}(k) + \sum_{\forall k \in C} QV_{at}(k)$$



〈그림 3.2〉 QV를 적용한 일정

*QV*를 *PV*와 *EV*를 함께 그래프에 표시하면 <그림 3.2>과 같이 표시된다. 누적 *QV*($\sum QV$)의 존재는 그만큼의 결함 제거 작업이 추가로 필요함을 나타낸다. $\sum QV$가 0이면 품질로 인한 추가작업이 필요 없음을 표시한다. 즉, 통합 테스트, 시스템 테스트, 인수 테스트에서 결함을 발견하고 수정할 가능성이 매우 적다는 것을 의미한다. 만일 $\sum QV$가 $\sum EV$보다 크다면 소프트웨어 작업은 단위 프로그램 개발작업에 사용한 작업량보다 더 많은 결함제거작업이 필요함을 추가로 나타낸다.

한편 *QV*를 활용하면 계획되어 있는 통합 테스트, 시스템 테스트와 인수 테스트 단계에 결함 제거를 위한 시간이 충분히 확보되어있는가를 판단할 수 있는데 이를 위한 지표로 *QV'* 을 도입한다.

먼저 통합 테스트 단계에 계획되어 있는 시간(PV_{it})에서 단위 프로그램 *k*가 차지하는 양($PV_{it}(k)$)는 전체 단위 프로그램에 계획된 작업량($\sum PV$)에서 단위 프로그램 *k*가 차지하는 비율(*EV*(*k*) 또는 *PV*(*k*))만큼 할당된다고 가정할 수 있다. $PV_{it}(k)$를 수식으로 표현하면 다음과 같다:

$$PV_{it}(k) = PV_{it} * PV(k) / \sum_{\forall j} PV(j)$$

$PV_{it}(k)$는 단위 프로그램 *k* 를 위해서 통합테스트 단계에서 사용할 수 있는 작업량인데, 통합 테스트 단계에서 결함 제거와 관련하여 추가적으로 필요한 $QV_{it}(k)$와 비교하여 통합 테스트 단계에서 결함 제거와 관련하여 충분한 시간이 계획되어 있는가를 예측할 수 있다. $QV_{it}'(k)$는 다음과 같이 정의된다:

$$\begin{aligned} QV'_{it}(k) &= QV_{it}(k) - PV_{it}(k) \\ &= QV_{it}(k) - PV_{it} * PV(k) / \sum_{\forall j} PV(j) \end{aligned}$$

그러므로 $QV'_{it}(k)$는 통합 테스트 단계에 단위 프로그램 k와 연관된 결함 제거 작업 시간이 충분히 계획되어 있는지를 판단하는 지표가 된다. $QV'_{it}(k)$가 0이면, 현재 계획되어 있는 통합 테스트 단계가 단위 프로그램 *k*로 인한 결함 제거 작업을 위해 충분함을 나타내며 계획이 비교적 정확하게 수립된 경우를 의미한다. 0보다 크면 계획된 통합 테스트 단계보다 더 많은 작업 시간이 필요함을 나타내며, 0보다 작으면 통합 테스트 단계가 원래의 계획보다 일찍 끝날 수 있음을 의미한다.

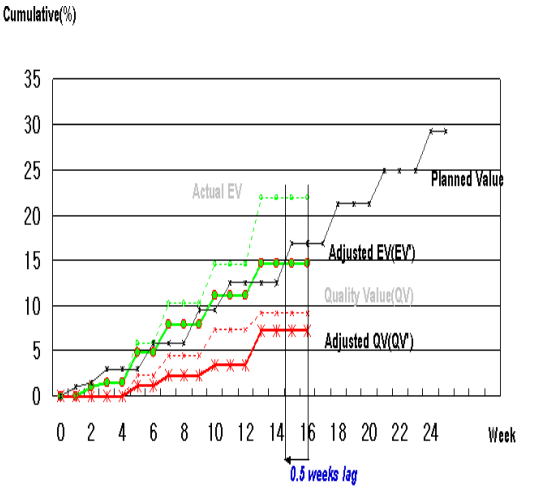
통합 테스트와 동일하게 시스템 테스트, 인수 테스트, 유지보수단계에서 *QV'* 를 구하면 다음과 같다:

$$QV'_{st}(k) = QV_{st}(k) - PV_{st} * PV(k) / \sum_{\forall j} PV(j)$$

$$QV'_{at}(k) = QV_{at}(k) - PV_{at} * PV(k) / \sum_{\forall j} PV(j)$$

$$QV'_{ad}(k) = QV_{ad}(k) - PV_{ad} * PV(k) / \sum_{\forall j} PV(j)$$

소프트웨어 일정 추적(EVT)을 할 때, *EV*값을 *QV'*로 보정한 Adjusted *EV*(*EV'*=*EV* - *QV'*)은 품질이 일정에 미치는 영향을 파악하는데 유용하게 활용될 수 있다.



〈그림 3.3〉 *QV'*를 적용한 일정 관리

일정 지연 여부를 단위 프로그램 개발 완료여부만으로 판단하는 <그림 2.1>에서는 4.3주 빨리 진행되는 것으로 보인다. 그러나 <그림 2.1>에 대해 *EV'*(Adjusted *EV*)을 적용해 보면 <그림 3.3>에서와 같이 예정보다 0.5 주 늦게 일정이 지연되고 있음을 예측할 수 있다.

Ⅳ. *QV*를 이용한 프로젝트 일정 예측

지금까지는 개발이 완료된 단위 프로그램들의 집합으로부터 품질을 고려한 작업 진척 상황을 추적할 수 있는 방법으로 *QV*와 *QV'*을 활용하였다. 이 결과를 확장하면 현 단계 이후의 전체 프로젝트 일정 추이를 품질과 관련하여 예측하는데 사용될 수 있다. 프로젝트 전체에서 예상된 *QV* 예상 값($QV_{project}$)과 단위 프로그램들의 *QV* 누적 값 ($\sum QV$)의 비율은 전체 단위 프로그램의 누적 *PV*값($\sum PV$)과 완료된 단위 프로그램의 누적 *EV* 값($\sum EV$)의 비율과 동일하다고 가정할 수 있다. 따라서 프로젝트 전체 통합 테스트에 필요한 품질지수($QV_{project,it}$)는 다음과 같이 유도할 수 있다:

$$QV_{project,\,it} = \sum_{\forall k \in C} QV_{it}(k) * \sum_{\forall j} PV(j) / \sum_{\forall k \in C} EV(k)$$

시스템 테스트와 통합 테스트에서 필요한 결함 제거 작업량도 통합 테스트 단계의 값 ($QV_{project,it}$)과 마찬가지로 구할 수 있다. 이를 이용하여 단위 테스트 이후 결함을 제거하는데 필요한 프로젝트 전체 품질 지수 *QV*를 다음과 같이 예측할 수 있다:

$$QV_{project} = QV_{project,\,it} + QV_{project,\,st} + QV_{project,\,at}$$

$QV_{project}$는 개발이 완료된 단위 프로그램의 집합으로부터 예측한 전체프로젝트의 품질지수가 된다. $QV_{project}$와 같이 $QV'_{project}$도 동일한 방법으로 표시할 수 있는데, 이 값은 단위 테스트 이후 단계에서 결함을 제거하는데 필요한 작업량이 프로젝트 계획에서 충분히 반영되었는지를 반영하는 지표로 사용할 수 있다:

$$QV'_{project} = QV'_{project,\,it} + QV'_{project,\,st} + QV'_{project,\,at}$$

$QV'_{project}$ 값이 0이 되는 경우, 프로젝트에 계획된 통합 테스트, 시스템 테스트, 인수 테스트 작업량이 적절함을 표시하고, 0보다 크면 추가 작업이 필요한 경우를, 0보다 작으면 계획보다 일찍 개발 완료될 수 있음을 의미한다.

이러한 예측 값들을 수시로 확인하고 추적함으로써 프로젝트 관계자들은 기술적인 효과보다는 자신들의 품질에 대한 이해와 기대로 호손 효과(Hawthorne Effect)를 기대 할 수 있다 (Parsons, 1974).

Ⅴ. *QV* 적용 방안

Ⅲ장과 Ⅳ장에서 소개한 *QV* 모델을 소프트웨어 개발 현장에서 이용하기 위해서는 필요한 데이터들이 체계적으로 수집 관리 되어야 한다. 본 논문의 모델을 처음 적용할 경우에는 결함 제거 효율(*y*)이나 결함 제거 시간(*e*)을 비롯한 값들에 대해 각 개발 팀에 대한 독자적인 자료가 없으므로 산업체 평균 값이나 추정 값을 사용할 수밖에 없으나 소프트웨어 프로젝트를 수행해 감에 따라 여러 프로젝트로부터 자료가 수집되면 더 정확한 값을 사용할 수 있을 것이다.

본 논문의 *QV* 적용을 위해 프로젝트의 각 수행 단계에서 수집해야 할 정보를 PSP/TSP 과정과 함께 <표 4.1>에 나타내었다.

PSP와 TSP를 따르고 있는 경우에는 <표 4.1>에 정리된 바와 같이 매주 혹은 규칙에 의해서 작업내용과 작업시간, 작업 중 발생된 결함 등을 수집할 수 있는 체계를 갖추게 된다. 또한 TSP를 적용하면 프로젝트를 3개월 단위로 반복 수행함으로써 더욱 효과적으로 자료를 수집할

수 있다. 본 연구의 모델은 PSP 엔지니어가 자신의 작업 과정에 TSP를 적용할 때 유용하게 이용될 수 있도록 되어 있는데, 만일 다른 개발 방법을 적용하고 있다면 이러한 자료를 수집하고 관리하기 위해서 <표 4.1>과 유사하게 단계별 결함정보를 수집하고 사용된 시간을 기록할 필요가 있다.

〈표 4.1〉 *QV* 계산을 위한 데이터[4)]

	PSP/TSP	최소정보	비고
준비단계	과거이력이나 산업통계자료에서 수집한 단계별 결함제거효율과 결함제거시간		
프로젝트계획	TSP launch, PSP1.1	단위프로그램 작업량산정, 일정 계획 수립	*PV* 준비
요구분석	PSP0.1		
상위단계설계	PSP0.1		
상위단계설계 검토	PSP2	(결함정보, 시간)	
상세단계설계	PSP2		단위테스트에서 결함정보와 시간정보가 핵심자료
상세단계설계 검토		(결함정보, 시간)	
코드 (프로그래밍)			
코드검토		(결함정보, 시간)	
컴파일		(결함정보, 시간)	
단위테스트		결함정보, 시간	
통합테스트	PSP1	결함정보, 시간	TSP 다음 프로젝트를 위한 자료수집
시스템테스트	PSP1	결함정보, 시간	
인수테스트	PSP1	결함정보, 시간	
유지보수	PSP2	결함정보, 시간	공급 후 6~12 개월

4) 표에서 () 부분은 선택사항을 의미함.

PSP/TSP의 적용여부에 관계없이 초기에는 프로젝트 계획 단계에서 세부 프로그램 작업을 목록화 하고 결함제거효율과 각 단계별 결함제거시간으로는 산업계표준 등을 사용하여 작업량과 작업일정을 계획함으로써 *PV*를 설정하는 것이 필요하다. 요구분석과 상세설계단계는 제거된 결함과 결함제거에 소요된 시간을 기록하는 것만으로도 충분하다. 그러나 상세설계에서부터 단위 테스트 단계까지는 개발자가 최소한 단위 테스트에 대한 결함만이라도 정확하게 기록해야 한다. 그 후의 단계들은 다음 프로젝트에 사용할 결함제거효율, 결함제거시간을 측정하기 위한 기초 자료를 수집하는 과정이다. PSP와 TSP를 사용하는 경우에는 PSP 과정에서 충분한 기본정보를 참조할 수 있기 때문에 별도의 자료수집을 정의하지 않아도 다음 프로젝트에 정확히 반영할 수 있는 결함제거효율과 결함제거시간이 산정될 뿐만 아니라 프로젝트 진행 도중에도 결함제거효율, 결함제거시간을 바로 적용할 수 있다.

본 논문의 모델을 적용하기 위한 프로젝트 크기도 고려되어야 한다. TSP를 적용하기 위한 프로젝트 규모는 2MM(Man Month) 이상 30MM 이하로 추천되므로(Humphrey, 2002) 본 논문의 모델을 적용하기 위해서는 대형 프로젝트인 경우 30MM이하로 분할하고 전체 프로젝트에 대해서 다시 통합하여 관리할 수 있는 모델도 필요하다.

VI. 결 론

소프트웨어 개발 일정은 결함 제거 시간과 밀접한 관계를 갖고 있다. 결함 제거를 위해 사용하는 결함 제거 방법은 일정한 결함 제거 효율을 가지고 있으며, 또한 결함 제거 시점에 따라 결함 제거 시간도 다르다. 본 논문에서는 단위 테스트 단계에서의 결함 정보를 사용하여 이후

단계에서 결함 제거를 위해 필요한 시간을 예측하기 위한 모델로 *QV*(Quality Value) 개념을 제안하였다. *QV*는 소프트웨어의 결함 제거를 위해 추가로 필요한 작업량으로 정의되는데, 이러한 결함제거작업량을 일정 관리에 도입함으로써 프로젝트 진행 중에 실시간으로 소프트웨어 품질로 인한 일정지연 여부를 계량화하여 확인 가능하도록 하였다. 본 논문의 모델을 사용함으로써 소프트웨어 프로젝트 관련자들은 소프트웨어 결함이 얼마나 많은 영향을 미칠 수 있는지 지속적으로 확인을 할 수 있게 됨으로써 심리적 효과를 얻을 수 있을 것으로 기대한다.

본 논문의 모델을 처음에 적용할 때는 산업계 평균값들을 사용해야 하지만, 개발 조직별로 소프트웨어 유형별로 결함제거효율과 결함제거시간 정보를 지속적으로 수집분석하면 더욱 정확한 결과를 기대할 수 있다. 즉, 본 연구의 모델에서는 개발자에 관계없이 각 단계에서의 결함 제거 효율과 결함 제거 시간이 일정한 값을 갖는다고 가정하였다. 그러나 본 논문의 모델은 각 개발자별로 결함제거효율과 결함제거시간의 차이가 있더라도 일정을 예측하도록 확장 적용할 수 있다.

추후에는 본 모델의 타당성을 검증하고 보완하기 위해서 실무 프로젝트에 본 논문의 모델을 적용하는 작업을 진행할 예정이다. 이를 바탕으로 통합 테스트 단계가 아니라 Review나 Inspection 등, 통합 테스트 단계보다 좀 더 이른 시점에서 *QV*를 계산할 수 있도록 하는 방안도 연구할 예정이다. 또한, 본 논문의 방안을 EVMS 모델에 통합하여 소프트웨어 프로젝트의 전체 과정을 일정, 비용, 품질, 위험 등을 종합적 관점에서 파악하도록 하는 방안에 대한 연구도 진행할 예정이다.

† 본 논문은 2004년 상명대학교 학술연구비 지원에 의해 연구되었음.

참 고 문 헌

장시영, 신동익, "정보시스템 성과평가 방법론 연구 - 개발프로젝트를 중심으로", 한국경영학회 경영저널, 제1권, 제1호, 2000, pp.189-207.

ANSI(American National Standards Institute), "ANSI-748: Earned Value Management Systems", 1998.

Basili, V. R., F. E. McGarry, R. Pajerski and M. V. Zelkowitz, "Lessons learned from 25 years of process improvement: the rise and fall of the NASA software engineering laboratory", *24th International Conference on Software Engineering*, May 2002, pp.69-79.

Chrissis, M. B., M. Konrad and S. Shrum, *CMMI: Guidelines for Process Integration and Product Improvement*, Addison Wesley, 2003.

Conn, R., "A Reusable, Academic-Strength, Metrics-Based Software Engineering Process for Capstone Courses and Projects", *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, March 2004, pp.492-946.

Curtis, B., W. E. Hefley and S. A. Miller, "People Capability Maturity Model (P-CMM)", *SEI Report*, CMU/SEI-2001- MM-01, July 2001.

Davis, N. and J. Mullaney, "The Team Software Process (TSP) in Practice: A Summary of Recent Results", *SEI Technical Report*, CMU/SEI-2003-TR-014, 2003.

Goldenson, D. R. and D. L. Gibson, "Demonstrating the Impact and Benefits of CMMI: An Update and Preliminary Results", *SEI Technical Report*, CMU/SEI- 2003-SR-009, 2003.

Humphrey, W. S., *A Discipline for Software Engineering*, Reading MA: Addison-Wesley, 1995.

Humphrey, W. S., *Introduction to the Personal Software Process*, Reading MA: Addison-Wesley, 1997.

Humphrey, W. S., *Introduction to the Team Software Process*, Reading MA: Addison-Wesley, 2000.

Humphrey, W. S., *Winning with Software: An Executive Strategy*, Reading MA: Addison-Wesley, 2002.

IFPUG(International Function Point Users Group), Function Point Counting Practices Manual, Version 4.1, 2003.

Jalote, P., *CMM in Practice: Processes for Executing Software Project at Infosys*, Reading MA: Addison-Wesley, 2000.

Jones, C., *Software Assessments, Benchmarks and Best Practices*, Reading MA: Addison-Wesley, 2000.

Krishnan, M. S., C. H. Kriebel, S. Kekre and T. Mukhopadhyay, "An empirical analysis of cost and conformance quality in software products", *Management Science*, Vol. 46, No. 6, June 2000, pp.745-759.

Liblit, B., A. Aiken, A. X. Zheng and M. I. Jordan, "Bug Isolation via Remote Program Sampling", *ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*, June 2003, pp.141-154.

Liu, X. F., G. Kane and M. Bambroo, "An Intelligent Early Warning System for Software Quality Improvement and Project Management", *15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI' 03)*, 2003.

Parsons, H. M., "What Happened at Hawthorne?", *Science*, Vol. 183, No. 8, Mar. 1974, pp. 922-932.

SEI(Software Engineering Institute), "Process Maturity Profile: CMMI v1.1 SCAMPI v1.1 Appraisal Results 2003 Year End Update", 2004, http://www.sei.cmu.edu./sema/pdf/CMMI/2004marCMMI.pdf.

Solomon, P., "From Performance-Based Earned Value (PBEV) to the Capability Maturity Model-Integrated (CMMI)", *DoD Software Technology Conference(STC)*, April 2002.

Zage, D. and W. Zage, "An Analysis of the Fault Correction Process in a Large- Scale SDL Production Model", *Proceedings of the 25th International Conference on Software Engineering*, May 2003, pp. 570-577.

SGI (Standish Group International Inc.), "Extreme chaos", 2001, http://standishgroup.com/sample_research/PDFpages/extreme_chaos.pdf.

Information System Review
Volume 6 Number 2
December 2004

A Suggestion for Merging Quality Management into Software Project Schedule Management

Seonuck Paek* · Yong-Soo Han** · Sugwon Hong***

Abstract

In this paper we propose a new software project development management model incorporating quality management to schedule management. Though many efficient techniques such as code review and inspection are used to remove defects, the effect of defect removal time on project schedule hasn't been studied much. However, poor quality management has an important effect upon overall schedule and sometimes software projects fails due to it. Thus, quality management and schedule management should be considered together and we need to reflect the time to maintain software quality into the schedule management. For the proposed model we introduced "Quality Value" representing the needed time to remove software defects. We assume PSP/TSP to gather the needed data for quality value. The proposed model can be used to predict the effect of software defects on schedule in advance and to prevent schedule lag.

Keywords: Quality Management, Schedule Management, PSP, TSP, Earned Value, Planned Value, EVMS, Quality Value

* Associative Professor, Dept. of Computer Software Engineering, Sangmyung University
** President, OSP Research Institute
*** Professor, Dept. of Computer Software, Myongji University

◐ 저 자 소 개 ◑

백 선 욱 (paeksu@smu.ac.kr)

서울대학교 컴퓨터공학과에서 학사, 석사, 박사 학위를 취득하고 현재 상명대학교 컴퓨터소프트웨어공학과 부교수로 재직하고 있다. 주요 관심 분야는 소프트웨어 프로젝트 개발 방법론, PSP/TSP, 전자상거래, 컴퓨터 네트워크 등이다.

한 용 수 (yshan@psptsp.com)

한국항공대학교 항공기계공학과에서 학사 학위를 취득하고, CMU에서 The Advanced Software Engineering Program for Korea를 이수하였다. 현재 OSP연구소 대표로 재직 중이며, PSP/TSP 포럼을 운영하고 있고 비트컴퓨터에서 소프트웨어 프로젝트관리 자문역을 맡고 있다. 프로젝트관리 표준을 활용하여 현대자동차, 인천제철, IMK 등의 구매/조달 시스템 개발을 관리하였으며 삼성전자, 한국은행, 서울시청 등 미들웨어 구축 컨설팅에 참여 하였다. 주요 관심분야는 정보화전략수립, S/W개발 프로젝트관리, 소프트웨어공학 등이다.

홍 석 원 (swhong@mju.ac.kr)

서울대학교 물리학과에서 학사 학위를 취득하고, North Carolina State University에서 전산학 석사 박사 학위를 취득하였다. KIST 소프트웨어 개발센터, 에너지 경제 연구원, SK㈜ 기술개발부, 전자통신연구원에서 근무를 했으며 현재 명지대학교 컴퓨터 소프트웨어학과 교수로 재직 중이다. 관심 분야는 시스템 성능 분석, 네트워크 기술 및 프로토콜, 시뮬레이션 등이다.

논문접수일 : 2004년 7월 9일 **게재확정일 : 2004년 9월 6일**
1차 수정일 : 2004년 8월 30일