# 결함 제거 시간을 반영한 소프트웨어 개발 일정 관리 방안

백선욱*, 한용수**

*상명대학교 컴퓨터소프트웨어공학전공, **OSP 연구소

# A Suggestion for Software Project Schedule Management Reflecting Defect Removal Time

Seonuck Paek*, Han YongSoo**

*Sangmyung University, **OSP Research Institute

## 요 약

본 논문에서는 일정한 소프트웨어 품질 수준을 달성하기 위해 소요되는 시간을 소프트웨어 개발 일정 관리에 반영한 새로운 일정관리 모델을 제안한다. 제안된 모델은 PSP/TSP (Personal Software Process/Team Software Process)의 일정 관리 및 추적 모델에 결함 제거에 필요한 작업량을 반영하여 품질 관리까지 포함한 종합적인 일정 추적과 관리가 가능하도록 하였다.

## 1. 서 론

Standish group의 2000년 보고서에 따르면 소프트웨어 개발 프로젝트에서 평균 전체일정의 63%에 이르는 일정지연이 발생하고[1], Capers Jones의 통계 자료에 의하면 소프트웨어 개발 프로젝트에서 결함 발견 및 제거 등 결함과 관련된 작업은 전체 작업 중에서 최대 36%까지 이른다[2]. 점차 대형화되는 소프트웨어 개발 작업의 많은 부분이 결함 발견 및 제거와 관계되어 있으며, 결함을 관리하는 방법에 따라서 일정지연을 사전에 상당 부분 예방할 수 있다[2,3].

따라서 소프트웨어의 결함이 일정에 미치는 영향을 참여자 스스로 정량적으로 확인함으로써 일정 지연을 최소화 하거나 일정을 단축할 수 있도록 지원할 수 있는 체계가 필요하다[6,7,8].

본 논문에서는 소프트웨어 개발 프로젝트의 진척 상황 관리를 위해 PSP1.1 에서 제공되는 일정 추적(EVT: Earned Value Tracking)[9]에 품질 요소를 함께 다룰 수 있도록 하는 방안을 제안하고자 한다. 이를 위해 본 논문에서는 품질 지수(QV :Quality Value)개념을 도입하여 프로그램 개발 과정에서 최종 제품의 결함 발생 가능성을 정량적으로 예측하고 이를 바탕으로 이러한 결함의 제거에 필요한 작업량이 일정에 어떠한 영향을 미칠지 분석할 수 있도록 하고자 한다. 본 논문의 연구 결과

는 프로젝트 개발 프로세스 진행 과정에서 일정, 비용, 위험 요소를 관리에 사용되는EVMS(Earned Value Management System)[7,8,10]에 통합되어 사용될 수도 있다.

본 논문의 2 절에서는 PSP/TSP의 품질 관리와 일정 관리에 대해 간략히 기술하고 3 절에서는 본 논문에서 제안한 통합 모델을 위한 QV 산정 모델 및 적용 방안에 대해 기술한다.

## 2. PSP/TSP 결함 관리와 일정 관리

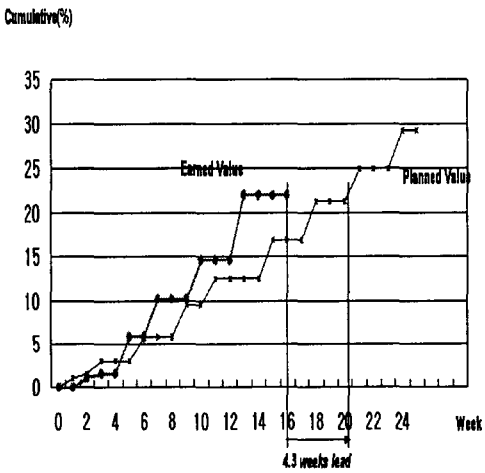
먼저, PSP/TSP의 결함 관리 체계와 일정 관리에 대해 서술한다. 결함 측정은 일정 크기의 프로그램마다 발견된 결함의 수로 나타내는데, defects / LOC LOC : Line of Code[1], 또는defects/FP[2] 로 표시된다. 일반적으로 인수 테스트(acceptance test) 방법에서는 전체 결함의 약 40% 정도만 발견할 수 있고 나머지 60% 정도는 공급 후에 발견되는 결함으로 밝혀져 있다[2]. 이러한 결함 발견 가능성을 "결함 제거 효율(Defect Removal Efficiency)"이라 하며 다음과 같이 표시 한다[2,9].

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

또한, 결함 제거 방법들의 효율성을 비교하기 위한 방법으로 하나의 결함을 제거하는데 필요한 작업량을 "결함 제거 시간"이라 하며 다음과 같이 나타낸다:

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

한편, PSP의 일정 계획(PSP1.1)에서는 작업 계획량을 PV(Planed Value)로, 작업 완료량을 EV(Earned Value)로 표시하여 전체 작업에서 진행된 작업량을 표시하는 척도로 이용한다. 이 때 사용하는 PV는 전체 프로그램을 개발하는데 필요한 작업에 대해 세부작업 항목의 작업량을 백분율로 표시하고, 세부작업이 완료되면 그 항목에 할당되었던 PV 값은 누적 EV에 합산된다. PSP에서는 해당 작업 항목이 완료 되었을 때만 그 항목에 할당된 PV를 EV에 적용되고 작업이 진행 중인 경우는 누적 EV에 합계되지 않는다(그림 1의 PV 및 EV 그래프 참조). 이러한 일정 관리를 이용함으로써 일정 준수 및 지연 여부를 파악할 수 있다.




[그림 1] PSP/TSP 일정 관리

## 3. 품질지수(QV: Quality Value)

2절에서 살펴보았듯이 품질 유지를 위한 결함 제거 작업은 일정에 밀접한 영향을 미치는데, 이러한 영향을 일정 관리에 함께 표시하는 방법으로 본 논문에서는 품질 지수(QV)를 제안한다.

먼저, 본 논문의 전개를 위해 먼저 소프트웨어 개발과정은 프로그램을 작은 단위로 분할하여 여러 개발자가 자신에게 분배된 단위 프로그램을 개발하고 점차 대형 소프트웨어로 통합하는 과정을 거친다고 가정한다. 본 논문에서 소프트웨어의 세부 기능에 대해 "상세 설계", "프로그래밍" 그리고 "

1) 프로그램소프코드의 라인수로 PSP에서 주로 이용됨[4,9].

2) FP : Function Point, 사용자 입장에서 프로그램의 크기를 산정하고 기술적. 환경적 측면을 고려한 크기 산정 방식[2,5].

단위 테스트"를 개발자가 반복적으로 하는 일련의 업무를 "단위 프로그램"이라고 한다.

QV는 품질이 일정에 미치는 영향을 표시하기 위한 지표로 단위 테스트(unit test) 단계에서의 결함의 수로 아직 제거되지 못한 결함제거에 필요한 예상 작업량을 예측하기 위해 사용된다.

먼저 단위 프로그램 k 의 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:

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

단위 테스트를 마치고 난 후의 통합 테스트(integration test) 시작 단계에서 단위 프로그램 $k$에 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:

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

단위 프로그램 $k$ 에 남아 있는 결함의 제거를 위해 통합 테스트 단계에서 필요한 추가 작업량($E_{it}(k)$)을 전체 결함의 수($Dt_{it}(k)$), 결함 제거 효율($y_{it}$) 그리고 결함 제거 시간($e_{it}$ )으로 다음과 같이 표시할 수 있다:

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

이 값은 결함 제거와 관련하여 통합 테스트 단계에서 추가로 필요한 작업량을 표시하고 있으므로 본 논문에서는 "결함 제거 작업량"이라 정의한다. 통합 테스트 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[2]. 위의 식들로부터 단위 테스트 단계에서 수정된 결함의 수 $Df_{ut}(k)$ 가 많을수록 단위 테스트 후에 결함 제거와 관련된 작업량이 많다는 것을 보여주고 있는데, 이러한 결과는 Watts Humphrey와 Capers Jones의 결과와 일치한다[2,4].

단위 프로그램 $k$ 와 관련한 결함 제거를 위해 통합 테스트 단계에서 추가로 필요한 작업량($E_{it}(k)$)를 EV와 함께 사용할 수 있도록 하기 위해서 EV와 동일한 단위 체계로 정규화 하여 QV로 나타내고자 한다. 통합 테스트 단계에서 단위 프로그램 k 와 관련하여 필요한 결함 제거 작업량($E_{it}(k)$)을 전체 프로젝트 개발 기간으로 정규화 하여 $QV_{it}(k)$라 하면 다음과 같이 나타낼 수 있다(여기서, $Ed(k)$ 는 단위 프로그램 $k$의 개발을 위해 계획되어 있는 시간이며, $PV(k)$ 는 $Ed(k)$ 를 전체 프로젝트 개발 시간으로 나눈 값으로 전체 프로젝트 개발 기간 중에서 단위 프로그램 $k$ 에 계획된 시간의 비율을 나타낸다):

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

일정 추적(Earned Value Tracking)에서 QV를 이용하기 위해서는 개발이 완료된 단위 프로그램 각각의 QV를 누적하여 사용한다. 단위 프로그램들 중에서 현재까지 개발이 완료된 것들의 집합을 C라 하자. 그러면 C에 속한 단위 프로그램과 관련된 결함 제거를 위해 통합 테스트 단계에서 다음의 값 만큼의 추가 작업이 필요함을 알 수 있다:

$$\sum_{\forall k \in C} QV_{it}(k)$$

이 값을 PV와 EV를 함께 그래프에 표시하면 그림 2와 같이 표시된다. 누적 QV(ΣQV)의 존재는 그만큼의 결함 제거 작업이 추가로 필요함을 나타낸다. ΣQV가 0이면 품질로 인한 추가작업이 필요 없음을 표시한다. 즉, 통합 테스트 단계에서 결함이 발견되고 수정될 가능성이 매우 적다는 것을 의미한다. 만일 ΣQV가 EV보다 크다면 소프트웨어

작업은 단위 프로그램 개발작업에 사용한 작업량보다 더 많은 결함 제거 작업이 추가로 필요함을 나타낸다.

한편 QV를 활용하면 계획되어 있는 통합 테스트 단계에 결함 제거를 위한 시간이 충분히 확보되어있는가를 판단할 수 있는데 이를 위한 지표로 $QV'$을 도입한다.

먼저 통합 테스트 단계에 계획되어 있는 전체 시간 ($PV_{it}$)에서 단위 프로그램 $k$가 차지하는 양 ($PV_{it}(k)$)는 전체 단위 프로그램에 계획된 작업량에서 단위 프로그램 $k$가 차지하는 비율($EV(k)$ 또는 $PV(k)$) 만큼 할당된다고 가정할 수 있다:

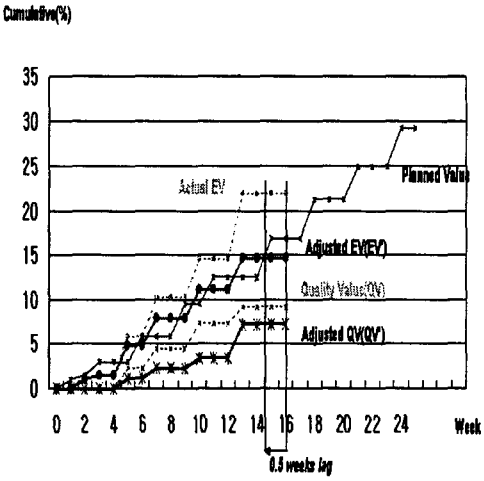
$$PV_{it}(k) = PV_{it} * PV(k) / \sum_{\forall j} PV(j)$$

$PV_{it}(k)$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. $QV'_{it}(k)$를 다음과 같이 정의한다:

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

그러므로 $QV'_{it}(k)$는 단위 프로그램 $k$와 연관된 결함 제거 작업 시간이 통합 테스트 단계에 충분히 계획되어 있는지를 판단하는 지표가 된다. $QV'_{it}(k)$가 0이면, 현재 계획되어 있는 통합 테스트 시간이 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.

소프트웨어 일정 추적(EVT)을 할 때, EV값을 QV'로 보정한 Adjusted EV(EV'= EV - QV')은 품질이 일정에 미치는 영향을 파악하는데 유용하게 활용될 수 있다. 그림 1의 예에서는 일정 지연 여부를 결함 제거를 위한 추가 시간을 고려하지 않고 단위 프로그램 개발 완료 만으로 추적하는 예를 보여주고 있는데, 계획보다 4.3 주 빨리 진행되고 있는 것으로 파악되고 있다. 그러나 그림 1에 대해 EV' (Adjusted EV)을 적용하여 본 그림 2에서 실제로는 예정보다0.5 주 늦게 일정이 진행되고 있음을 알 수 있다.




[그림 2] QV'를 적용한 일정 관리

본 논문에서 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. 만일 개발 과정에 PSP/TSP를 적용하고 있다면 결함 관련 정보 및 각 개발 단계별 개발 시간 정보가 쉽게 확보되므로 본 논문의 모델을 보다 쉽게 적용할 수 있다.

## 4. 결론 및 토론

본 논문에서는 기존의 일정 관리(EVT: Earned Value Tracking) 시스템에서 소프트웨어 결함을 제거하기 위해 필요한 시간을 반영한 일정 관리를 위해 QV(Quality Value)를 새로 제안하였다. QV는 소프트웨어의 결함 제거를 위해 추가로 필요한 작업량으로 이러한 결함 제거 작업량을 일정 관리에 도입함으로써 소프트웨어 품질까지 고려한 일정 관리가 가능하도록 하였다.

정확한 QV를 산정하기 위해선 조직별, 소프트웨어 유형별로 결함제거효율과 결함제거시간 정보를 지속적으로 수집분석하면 더욱 정확한 결과 기대할 수 있다. 비록 초기 적용에 있어서 정확하지 않은 결과를 제공한다고 하더라도 소프트웨어 프로젝트 관련자들은 소프트웨어 결함이 얼마나 많은 영향을 미칠 수 있는지 지속적으로 확인을 할 수 있게 됨으로써 심리적 효과를 같이 얻을 수 있을 것으로 기대할 수 있다. 특히 개발 과정에 PSP/TSP를 적용하고 있다면 본 논문의 모델 적용에 필요한 데이터는 보다 쉽게 얻을 수 있으나 그렇지 않은 경우에도 별도로 필요한 데이터를 수집한다면 본 논문의 모델은 쉽게 적용 가능 하다.

본 논문에서 제시된 모델의 타당성을 검정하고 보완하기 위해서는 실제 프로젝트에 본 논문의 모델을 적용하여 검증하는 작업이 추가로 필요하다. 이를 바탕으로 통합 테스트 단계가 아니라 Review나 Inspection 등, 통합 테스트 단계보다 좀 더 이른 시점에서 QV를 계산할 수 있도록 하는 방안도 연구할 예정이다. 또한, 본 논문에서는 통합테스트 단계에서 필요한 결함 제거 작업량 산정까지만 추정하였으나, 그 이후의 시스템 테스트, 인수 테스트, 인수 후 유지 보수 단계의 테스트 과정까지 확장하는 것도 가능한데, 이러한 단계까지 포함하는 모델 개발도 진행할 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.

[참고문헌]


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

[2] Capers Jones, Software Assessments, Benchmarks, and Best Practices, Addison-Wesley, 2000.

[3] Noopur Davis et al., "The Team Software Process (TSP) in Practice: A Summary of Recent Results", Technical Report, CMU/SEI-2003-TR-014, 2003.

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

[5] Software Engineering Institute, "Process Maturity Profile", 2004(http: //www.sei.cmu.edu/sema/pdf/CMMI/ 2004marCMMI.pdf).

[6] Pankaj Jalote, "CMM in Practice: Processes for Executing Software Project at Infosys", Addison-Wesley, 2000.

[7] Paul Solomon, "From Performance-Based Earned ValueSM (PBEVSM) to the Capability Maturity Model-Integrated (CMMISM)", DoD Software Technology Conference, 2002.

[8] 장시영, 신동익, "정보시스템 성과평가 방법론 연구 - 개발프로젝트를 중심으로," 경영저널 1권 1호, 189-207, 2000.

[9] Watts S. Humphrey. "A Discipline for Software Engineering," Reading, MA: Addison-Wesley, 1995.

[10] American National Standards Institute, Earned Value Management Systems, ANSI-748, 1998.