소프트웨어 시험 인력의 추정 방법

박 주 석†

요 약

성공적인 프로젝트 계획은 활용 가능한 일정과 더불어 프로젝트를 완수하는데 요구되는 인력을 얼마나 정확히 추정하느냐에 달려있다. 새로운 또는 보다 나은 모델 개발에 많은 연구가 이루어졌지만 현존하는 소프트웨어 인력 추정 모델들은 소프트웨어 생명주기 전반에 걸쳐 투입되는 총 개발인력 또는 시간에 따른 단위시간당 개발인력 분포만을 제공하고 있다. 본 논문은 시간의 함수가 아닌 발견된 결함 수에 따라 시험단계에 투입되는 인력과 시험단계의 시험과정과 디버깅과정에 투입되는 인력을 추정하는 모델들을 제시하였다. 시험과 디버깅 과정에서 발견되는 결함에 기반한 투입 인력의 다항식 모델을 제안한다. 제안된 모델들은 5종의 다른 소프트웨어 프로젝트들에 적용되어 결정계수와 MMRE를 통해 모델의 적합성을 검증하였다.

An Estimating Method for Software Testing Manpower

Ju-Seok Park†

ABSTRACT

Successful project planning relies on a good estimation of the manpower required to complete a project, together with the schedule options that may be available. Despite the extensive research done developing new and better models, existing software manpower estimation models are present only the total manpower or instantaneous manpower distribution according to the testing time for the software life-cycle. This paper suggests the manpower estimating models for software testing phase as well as testing process and debugging process in accordance with detected faults. This paper presents the polynomial model for effort based on testing and debugging faults. These models are verified by 5 different software project data sets with coefficient of determination and mean magnitude of relative error for adaptability of model.

키워드 : 인력(Manpower), 시험단계(Testing Phase), 시험과정(Testing Process), 디버깅과정(Debugging Process), 통계적 모델(Statistical Model)

1. 서 론

대형 소프트웨어 프로젝트의 1%만이 계획된 기간과 예상 비용한도 내에서 고객을 만족시키며 완료되었는데 반해, 대부분의 프로젝트들은 1년 이상의 일정이 지연되고 초기 예상 비용의 2배 정도가 초과되었다[1]. 이와 같은 이유로 인해, 프로젝트 관리 측면에서 소프트웨어 개발 및 유지보수 비용을 줄이고자 체계적인 연구가 수행되고 있으며, 소프트웨어 비용산정 및 프로젝트의 일정관리 모델을 개발하는 계기가 되었다. 계획단계에서 보다 정확한 비용산정 및 일정관리는 프로젝트를 관리할 때 발생되는 다양한 의사결정, 소요 예산 및 개발인원 할당과 계약체결 여부에 신뢰할 만한 정보를 제공한다.

소프트웨어 개발에 투입되는 총 인력의 규모를 추정하는 모델로는 LOC(Line Of Code)를 이용한 Boehm의 COCOMO(COnstructive COst MOdel) 모델[2, 3] 등과 기능점수(FP, Function Point)를 이용한 Albrecht[4, 5], Kemerer[6], Matson[7] 등이 있다. LOC를 이용한 모델은 사용되는 언어에 종속되어 있으며, 코딩이 종료된 시점에서 정확한 LOC(Line Of Codes)를 측정할 수 있는 단점이 있는 반면에 기능점수를 이용하는 모델은 언어에 독립적이며, 소프트웨어 개발 초기인 요구사항 명세(Requirement Specification) 단계에서 개발인력과 비용을 추정할 수 있어 LOC를 이용한 모델의 단점을 보완할 수 있는 방법중 하나이다. 이와 같은 연구들은 모두 소프트웨어 프로젝트의 생명주기 전반에 걸쳐 투입되는 총 개발인력과 비용만을 측정하는 것이며, 생명주기의 각 단계별로 투입되는 인력의 규모, 시간에 따른 투입 인력의 규모 변화를 측정할 수 없다.

이에 반해, 소프트웨어 생명주기 전체와 생명주기의 각 단계인 요구사항 분석, 설계, 코딩 , 시험과 유지보수 단계에서 투입되는 인력이 시간에 따라 어떤 분포를 따르는지를 연구한 사례는 Putnam[8]의 SLIM(Software LIfe cycle Management) 모델을 비롯한 많은 모델들이 있다. 150개의 소프트웨어 프로젝트가 Norden[12]과 Putnam[8]에 의해 연구되

† 정 회 원 : 국방대학교 직무연수부 교수
논문접수 : 2004년 9월 7일, 심사완료 : 2004년 9월 30일

었으며, 소프트웨어 생명주기 전반에 걸쳐 소요되는 총 인력뿐만 아니라 생명주기의 각 단계인 요구명세 분석, 설계, 코딩, 시험 및 유지보수 등에 투입되는 인력도 Rayleigh 분포를 따름을 관찰하여 SLIM 모델을 제안하였다. 또한 Yamada et al.[9, 10]과 Pillai et al.[11]에 의해 지수, 와이블과 감마분포를 따르는 모델들이 제안되었다. 그러나 이들 모델을 적용하더라도 소프트웨어를 개발하는데 투입되는 총 인력의 규모를 생명주기의 각 단계별로 얼마로 할당할 것인가에 대한 연구 결과가 없어 각 단계별로 소요되는 인력을 추정할 수 없는 문제점을 갖고 있다.

소프트웨어 시험단계에 투입된 인력을 고찰해 보자. 시험단계는 결함의 원인이 되는 결함을 발견하기 위한 시험과정과 발견된 결함을 제거하는 디버깅 과정이 동시에 수행되는 특징을 갖고 있다. 또한, 시험단계에 투입된 인력 분포는 Rayleigh, Exponential, Weibull이나 Gamma 분포를 따르지 않는 경우가 빈번히 발생한다. 따라서, 시험단계에 투입되는 인력이 제안된 어떤 분포를 따르거나 따르지 않는 모든 프로젝트들에 적합하면서도 시험과 디버깅 과정 각각에 대해 투입되는 인력 추정 모델뿐만 아니라 시험과 디버깅 과정 모두에 투입되는 총 투입인력을 추정하는 모델이 필요하다. 따라서, 본 논문은 실제 개발된 프로젝트들의 정보를 근거로 소프트웨어 개발 각 단계별로 최적의 인력을 할당할 수 있는 기준을 제시하고자 한다.

2장에서는 기존의 개발인력 할당비율에 관한 연구들을 살펴보고, 3장에서는 실제 수행된 프로젝트들의 사례를 통해 시험과 디버깅 과정에서 투입되는 인력을 추정하는 모델을 제시한다. 이어서 4장에서는 시험단계에 투입되는 인력을 추정하는 모델을 제시한다.

2. 기존 연구 및 연구배경

LOC를 이용한 소프트웨어 개발인력 추정 모델은 식 (1) 또는 식 (2)의 형태를 취한다. 식 (1)과 식 (2)에서 E는 예측된 개발인력으로 Man-Months로 측정되고, a, b, c는 상수이며, KLOC는 최종 코딩된 소프트웨어의 추정된 KLOC (Thousands of Line Of Code)의 수이다. 예로 Bailey-Basili 모델은 $E = 5.5 + 0.73 \times (KLOC)^{1.16}$, Walston-Felix 모델은 $E = 5.2 \times (KLOC)^{0.91}$, Boehm의 Complex 모델은 $E = 2.8 \times (KLOC)^{1.20}$이다.

$$E = a + b \times (KLOC)^c \quad (1)$$

$$E = a \cdot (KLOC)^b \quad (2)$$

FP를 이용한 소프트웨어 개발인력 추정 모델은 식 (3)~식 (6)의 형태를 취한다. Albrecht et al.[4, 5] 모델은 $E = -13.39 + 0.0545FP$와 $\sqrt{E} = 1.000 + 0.00468FP$, Kemerer[6] 모델은 $E = -121.57 + 0.3411FP$, $E = -69.13 + 0.723FP - 8.054 \times 10^{-4}FP^2 + 3.073 \times 10^{-7}FP^3$와 $E = 60.62 + 7.728 \times 10^{-8}FP^3$, Matson et al.[7] 모델은 $E = 585.7 + 15.12FP$와 $\ln(E) = 2.51 + 1.00\ln(FP)$이 있다.

$$E = a + b \cdot FP \quad (3)$$

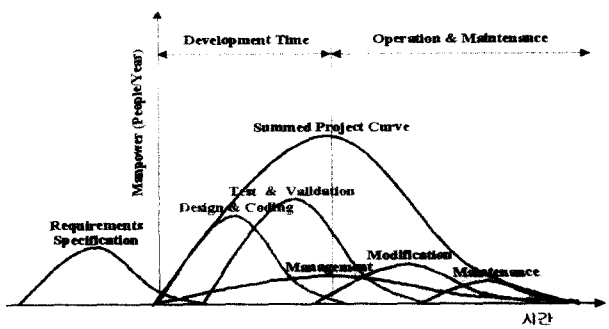
$$\sqrt{E} = a + b \cdot FP \quad (4)$$

$$E = a + b \cdot FP + c \cdot FP^2 + d \cdot FP^3 \quad (5)$$

$$\ln(E) = a + b \cdot \ln(FP) \quad (6)$$

그러나 LOC나 FP를 이용한 개발인력 추정 모델들은 소프트웨어 생명주기 전체에 걸쳐 투입되는 총 개발인력의 규모만을 추정할 수 있으며, 시간에 따른 개발인력의 규모 변화나 개발 각 단계별로 투입되는 인력 규모를 추정할 수 없는 단점을 갖고 있다.

소프트웨어 생명주기 (Life Cycle) 전반에 걸쳐 투입되는 인력 (Manpower)의 변화에 관한 연구로 Boehm[2], Putnam[8], Yamada et al.[9, 10], Pillai et al[11]와 Nordan[12, 13] 등이 있다. Norden[12]은 IBM에서 개발된 다양한 하드웨어 개발과정에 투입된 인력은 근사적으로 Rayleigh 분포를 따른다는 사실을 관찰하였다. 이후 Putnam[8]에 의해 이 관찰 결과가 소프트웨어 프로젝트에 적용되었다. 150개의 소프트웨어 프로젝트가 Norden[13]과 Putnam[8]에 의해 연구되었으며, 소프트웨어 생명주기 전반에 걸쳐 소요되는 총 인력뿐만 아니라 생명주기의 각 단계 (즉, 요구명세 분석, 설계, 코딩, 시험 및 유지보수 등)에 투입되는 인력도 (그림 1)과 같이 Rayleigh 분포를 따름을 관찰하였다. 임의의 시간 t 시점까지 프로젝트에 투입된 누적 인력 y를 식 (7)로, 임의의 시간 t 시점에서 프로젝트에 투입되는 인력 $\dot{y}$는 식 (7)을 미분한 식 (8)로 표현된다.



(그림 1) 소프트웨어 개발인력 분포

$$y = E(1 - e^{-at^2}) \text{ MY} \quad (7)$$

$$\dot{y} = 2Eate^{-at^2} \text{ MY/YR} \quad (8)$$

여기서 E는 Rayleigh 곡선 아래에 있는 영역으로 년 인원으로 표시되며, 소프트웨어 생명주기 전반에 걸쳐 투입된 총 인력이다. $a = 1/2t_d^2$이며, t_d는 $\dot{y}$이 최대가 되는 시점으로 경험적으로 볼 때, 시스템이 운영되는 시점에 근접한다. 따라서, $t_{\dot{y}_{max}} = t_d$를 시스템의 개발기간이라 하며, 소프트웨어 생명주기 전반에 걸쳐 총 소요되는 인력의 약 40%에 도달

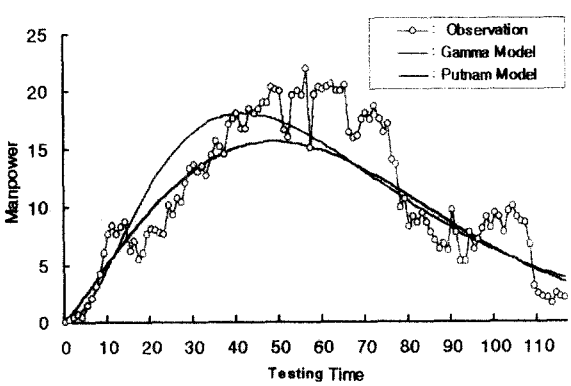
하는 시점이다. $a = 1/2t_d^2$로 치환하면, y는 식 (9)가 된다.

$$y = \frac{E}{t_d^2} te^{-\frac{t^2}{2t_d^2}} \tag{9}$$

Yamada et al.[9]는 소프트웨어 총 개발인력의 약 40~50%가 시험단계에 투입되며, 소프트웨어 시험 중 투입되는 시험 인력과 소프트웨어 생명주기에 투입되는 총 인력의 규모가 Rayleigh 분포를 따르는 모델을 제안하였으며, Rayleigh 분포의 대안으로 Exponential 분포를 제안하였다. Yamada et al.[10]은 실제 투입된 시험 인력에 대해 Exponential이나 Rayleigh 분포로 표현하는데 많은 문제점이 있음을 지적하여 소프트웨어 시험 중 결함 발견율은 단위시간에 투입되는 시험 인력의 규모에 비례한다는 가정하에 시험인력의 규모를 추정하는데 Weibull 분포를 따르는 모델을 제안하였다. 또한, 이상운[14]은 시험단계의 각 시점에 기반하여 인력이 어떻게 투입되는가에 대한 시그모이드 인력분포 모델을 제안하였다.

Boehm[2]은 소프트웨어 개발 초기에 느린 개발인력 투입 곡선 부분과 마지막 단계에서의 점차 길게 감소하는 Rayleigh 분포 특성이 대부분의 소프트웨어 프로젝트의 개발인력 곡선에 일치하지 않음을 지적하였다. 일반적으로 소프트웨어 프로젝트는 개발 초기에 하드웨어 프로젝트보다 빠른 투입인력 형성 비율(Buildup Rate)을 가지며, 이는 Rayleigh 분포와 편향된 결과를 나타낸다. 이 연구 결과를 토대로, Pillai et al.[11]은 Putnam[8]의 개발인력 분포가 Rayleigh 분포를 따르는데 따른 문제점을 해결하기 위해 (그림 2)의 FORTRAN 언어로 작성된 1,000 LOC 규모의 200개 모듈로 구성된 실시간 제어와 감시 프로그램에 대한 데이터를 이용하여 식 (10)의 Gamma 모델을 제시하였다.

$$y = \frac{8E}{\Gamma(3)t_d^3}(t^2 e^{-\frac{2}{t_d}t}) = \frac{4E}{t_d^3} t^2 e^{-\frac{2}{t_d}t} \tag{10}$$

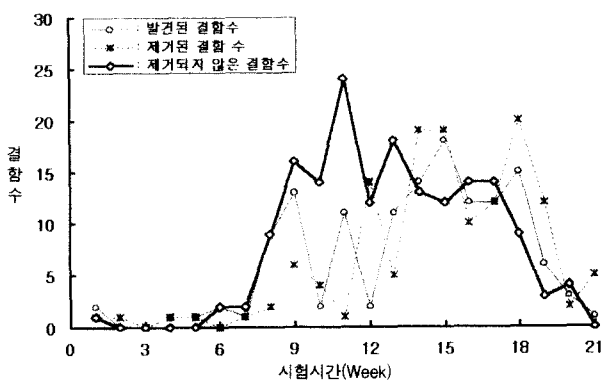


(그림 2) 개발인력 분포 사례

이들 제안된 모델들을 이용하면 소프트웨어 프로젝트의 일정한 시점에서 투입된 인력과 총 인력을 산정할 수 있어 사업초기에 의사결정 자료로 활용이 가능하다.

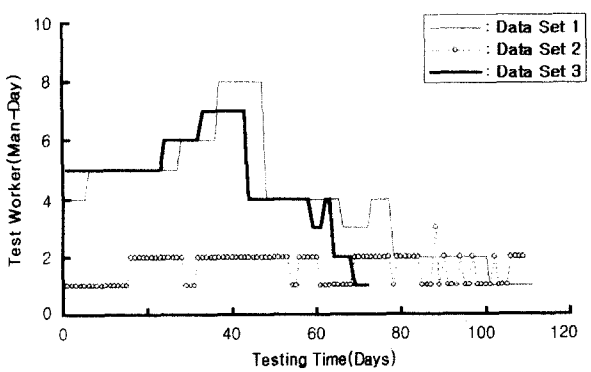
이상과 같은 연구 결과를 토대로 할 때, 개발의 각 단계별로 투입되는 인력도 일반적으로 Rayleigh, Exponential, Weibull 또는 Gamma 분포를 따라야만 한다. 그러나 투입되는 인력이 이들 분포를 따르지 않는 경우가 빈번히 발생하며, 시험단계에 투입되는 인력은 결함의 원인이 되는 결함을 발견하기 위한 시험과정과 발견된 결함의 원인이 되는 결함을 제거하기 위한 디버깅과정에 투입되는 인력이 요구되는 특성을 갖고 있다.

소프트웨어 신뢰성 모델은 일반적으로 발견된 결함은 즉시 제거되고 결함을 제거하는 과정에서 새로운 결함이 추가되지 않는다는 완전 디버깅(Perfect Debugging) 또는 새로운 결함이 추가될 수 있다는 불완전 디버깅(Imperfect Debugging)의 가정을 토대로 모델을 제시하고 있다. 그러나, (그림 3)의 사례와 같이 시험과정에서 발견된 결함은 즉시 제거되지 않는다. 그림에서 임의의 시험시간을 기준으로 하였을 경우, 발견된 결함 수 보다 제거된 결함 수가 많은 경우가 있다. 이는 이전 시간에 발견되어 제거되지 않고 남아 있는 결함을 해당 시점에서 제거함으로 인해 발생하는 현상이다.

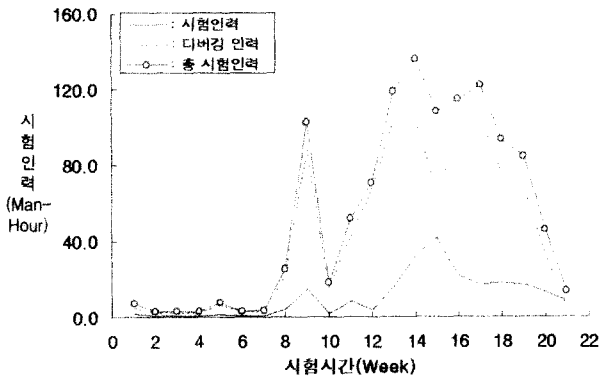


(그림 3) 시험단계에서 결함 발견과 제거 과정 사례

소프트웨어 시험단계에 투입된 인력에 관해 (그림 4)와 (그림 5)의 4개의 데이터를 고찰해보자.



(그림 4) 시험단계 투입 인력(Data Set 1~3)



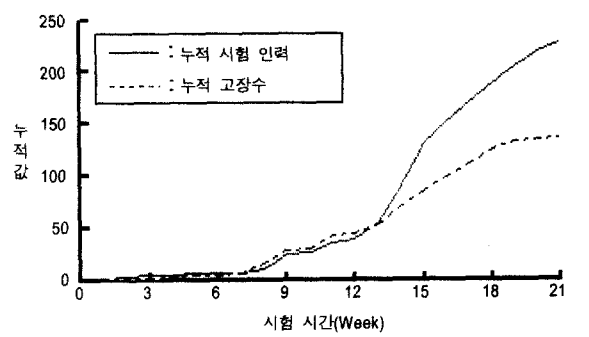
(그림 5) 시험단계 투입인력 (Data Set 4)

4개의 데이터를 각각 Data Set 1, Data Set 2, Data Set 3와 Data Set 4라 하자. Data Set 1은 Tohma, Jacoby, Murate와 Yamamoto[15]의 Table 1 데이터로 111일 동안 운영시험을 한 결과 481개의 결함이 발견되었으며, 이에 투입된 인력은 482명이다. Data Set 2는 Minohara와 Tohma [13]의 Table 9 데이터로 111일 동안 481개의 결함이 발견되었으며, 시험과 디버깅과정에 투입된 인원은 442명이다. Data Set 3은 Minohara와 Tohma[16]의 Table 10 데이터로 109일 동안 535개의 결함을 발견하였으며, 시험과 디버깅 과정에 174명이 투입되었다. Data Set 4는 실시간 제어 시스템인 Musa, Iannino와 Okumoto[17]의 21,000 라인의 System T1 프로그램으로 21주 동안 결함 식별과 제거에 1141시간이 투입되었다.

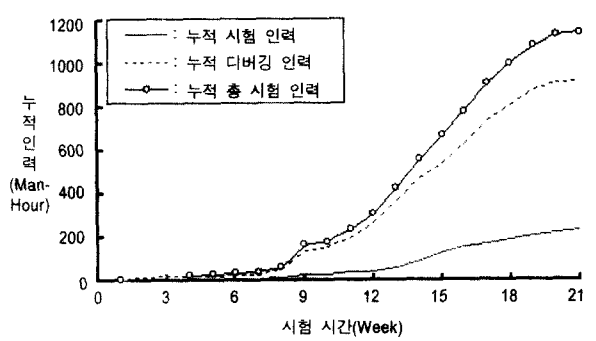
4개 Data Set들의 투입인력 분포를 살펴보면 Data Set 1, Data Set 3와 Data Set 4가 개략적으로 어떤 분포를 따르며, Data Set 2는 이들 분포를 전혀 따르지 않음을 알 수 있다. 또한 Data Set 1과 Data Set 2는 시험 시작 초반부에 많은 인력이 투입되어 Rayleigh, Exponential, Weibull이나 Gamma 분포와 편향된 결과를 보이며, Data Set 4는 시험 초반부에 이들 분포를 따르지 않다가 중반부 이후에 이들 분포를 따른다고 할 수 있다. 따라서, 이들 데이터들에 Rayleigh, Exponential, Weibull이나 Gamma 분포를 이용해 투입되는 개발인력을 추정하면 많은 편차를 가진 결과를 얻을 수 있으며, 시험단계에 투입되는 보다 정확한 인력 추정 모델이 필요함을 알 수 있다. 즉, 소프트웨어 시험단계에서 투입되는 인력은 시험과 디버깅 과정에 투입되는 인력을 모두 고려되어야만 한다. 시험의 목적은 결함을 많이 발견하는 것이며, 되도록 많은 결함을 발견하기 위해서는 많은 인력이 필요하며, 발견된 결함의 수에 따라 디버깅 과정에 소요되는 인력도 함께 증가할 수 있다. 따라서, 발견되는 결함 수와 투입 인력의 규모 사이에 특정한 관계가 있을 수 있으며, 이 경우에 대해 투입되는 인력을 추정할 수 있는 모델 개발로 시험단계에 투입될 인력의 분포를 추정하여 프로젝트 계획 단계에서 인력할당 문제에 대한 의사결정 자료로 활용될 수 있을 것이다. 따라서, 기존 Putnam[8]의 SLIM 모델은 투입 인력이 시간에 대한 함수로 표현되는 반면 본 제안 모델은 투입 인력이 발견된 누적 결함 수에 대한 함수로 표현되는 차이점이 있다.

3. 시험과 디버깅 과정 투입인력 추정 모델

Data Set 4를 고찰해보자. Data Set 4에서 시험과정에서 발견된 결함 수와 시험에 투입된 인력, 발견된 결함을 제거하는 디버깅에 투입된 인력 분포가 유사한 경향을 보이고 있다. 그러므로 발견되는 결함 수를 이용해 결함을 발견하는데 투입된 시험 인력과 발견된 결함을 제거하는데 투입된 인력을 추정하는 모델을 제시할 수 있다. 즉, 발견된 결함의 수에 따라 투입되는 시험 인력과 디버깅 인력의 규모에 영향을 미침을 알 수 있다. 발견된 결함 수와 시험과 디버깅에 투입된 인력에 대한 누적 분포는 (그림 6)과 (그림 7)에 제시되어 있다. 이들 분포로부터 투입되는 인력을 추정하는 모델 유도가 가능해진다.



(그림 6) Data Set 4의 누적 시험인력과 누적 결함 수 분포



(그림 7) Data Set 4의 누적 시험과 누적 디버깅 인력 분포

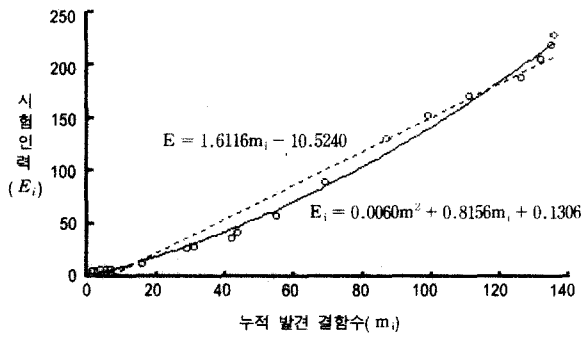
3.1 시험과정 투입인력 추정 모델

Data Set 4의 누적 결함 수에 따른 누적 시험인력 관계는 (그림 8)에 제시되어 있다. 누적 시험 투입 인력을 E_i, 누적 발견 결함 수(Cumulative number of identified faults)를 m_i라 하자. 그림에서 이들 관계는 선형보다는 곡선 형태를 갖고 있다. 누적 발견 결함 수와 결함을 발견하기 위한 시험

과정에 투입되는 인력이 선형이 아니고 비선형 형태로 증가하는 곡선 형태를 취하고 있다. 이는 소프트웨어에 남아 있는 결함이 시험과정에서 계속적으로 발견되고 디버깅 과정을 거치면서 제거되어 남아 있는 결함이 점차 감소한다. 이로 인해 결함을 발견할 수 있는 확률이 점차 감소하며, 이에 따라 시간이 지나면서 동일한 결함 수를 발견하는데 소요되는 인력은 증가하기 때문이다. 따라서, 시험에 투입되는 누적 인력 E_i는 식 (11)로 표현될 수 있다. 제안된 모델의 모수들은 회귀분석을 거쳐 추정되었으며, 식 (12)의 모델을 얻었다.

$$E_i = \alpha_1 m_i^2 + \beta_1 m_i + \gamma_1 \quad (11)$$

$$E_i = 0.0060\, m_i^2 + 0.8156 m_i + 0.1306 \quad (12)$$



(그림 8) 누적 결함 수와 누적 투입 인력 관계

추정된 비선형 모델의 회귀분석 결과 상관계수는 0.9970, 표준오차는 6.5741을 얻었다. 분산분석 결과는 <표 1>에, 제시되어 있다. 제안된 모델의 유의성 검정결과 $F \geq F(\alpha) = 3,128.66 > 1.49E-22$로 회귀 분석이 유의함을 알 수 있다.

〈표 1〉 분산분석

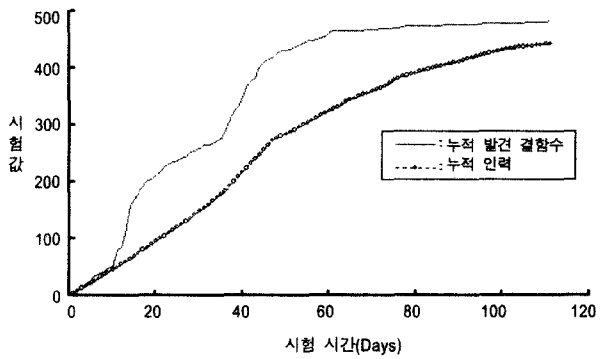
요인	제곱합	자유도	제곱평균	F	F(α)
회귀	135,216.23	1	135,216.23	3,128.66	1.49E-22
잔차	821.15	19	43.22		
계	136,037.38	20			

제안된 모델의 성능은 결정계수(Coefficient of determination, R^2)와 MMRE (Mean Magnitude of Relative Error)로 평가되었다. 결정계수는 제안된 모델이 실제 데이터의 변동을 얼마나 잘 표현할 수 있는지를 평가하는 기준이다. 모델 정확도를 평가하는 측도인 MMRE는 다음과 같이 측정된다. E = (추정치 − 실측치) /실측치, RE = |RE|, MMRE (Mean MRE) = $100/n * \sum_{i=1}^{n} MRE$. MMRE가 작은 값이면 모델은 평균적으로 좋은 모델임을 알 수 있다. <표 2>에는 선형모델과 제안된 비선형 모델의 성능을 비교 제시하였다. 제안된 비선형 모델이 결정계수가 보다 좋으면서 평균상대오차도 적은 결과를 나타내고 있음을 알 수 있다.

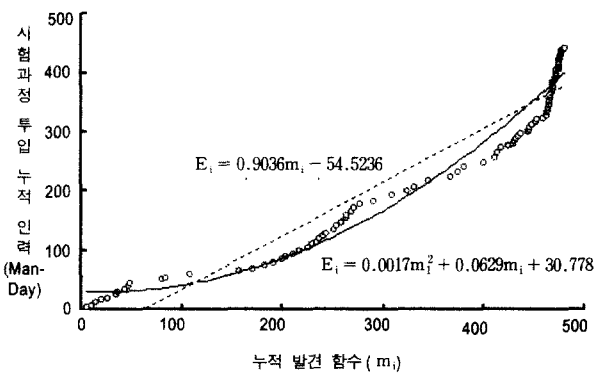
〈표 2〉 Data Set 4의 누적 투입인력 모델 성능 비교

인력 추정 모델	성 능	
	결정계수	MMRE
$E_i = 0.0060 m_i^2 + 0.8156 m_i + 0.1306$	0.9940	19.49%
$E_i = 1.6116 m_i - 10.5240$	0.9786	87.68%

추가적으로, (그림 9)와 같이 Data Set 1에 대해 시험시간에 따른 누적 결함 수와 누적 디버깅 인력 분포와 누적 결함 수와 누적 디버깅 투입 인력간 관계로부터 유도된 모델을 표현하였다. <표 3>에서 제안된 비선형 모델이 결정계수가 보다 좋으면서 평균상대오차도 적은 결과를 나타내고 있어 제안 모델이 보다 적합함을 알 수 있다.



(a) 시험시간에 따른 누적 시험 인력과 발견된 결함 수



(b) 누적 발견 결함 수에 따른 누적 시험 투입인력

(그림 9) Data Set 1의 누적 시험과정 투입인력 추정 모델

〈표 3〉 Data Set 1의 누적 시험인력 모델 성능 비교

인력 추정 모델	성 능	
	결정계수	MMRE
$E_i = 0.0017 m_i^2 + 0.0629 m_i + 30.7780$	0.9656	19.61%
$E_i = 0.9036 m_i - 54.5236$	0.9162	49.65%

3.2 디버깅과정 투입인력 추정 모델

Data Set 4의 누적 디버깅 결함 수에 따른 누적 디버깅 결함 수와 누적 디버깅 투입 인력 분포는 (그림 10)(a)에 제

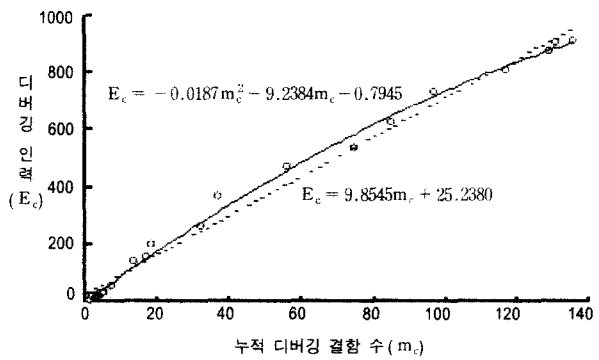
시되어 있다. 누적 디버깅 투입 인력을 E_c, 누적 디버깅 결함 수(Cumulative number of corrected faults)를 m_c라 하자. 그림에서 이들 관계도 시험과정에 투입되는 인력과 동일하게 선형보다는 곡선 형태를 갖고 있다. 디버깅 과정을 거치면서 숙련도와 소프트웨어에 대한 이해력이 점차 향상됨으로써 디버깅 과정에 투입되는 인력은 디버깅될 결함 수에 대해 점차 증가하다가 감소하는 경향을 나타낼 수도 있으며, 또한 이와 반대로 나중에 발견되는 결함을 제거하는데 어려운 문제점에 부닥쳐서 인력 규모가 점진적으로 증가하는 경향을 나타낼 수도 있다. 그러나 이들 현상 모두 선형보다는 곡선 형태의 관계로 디버깅 투입 인력을 추정 할 수 있다. 또한 발견된 누적 결함 수를 제거하기 위해 투입되는 디버깅 인력의 규모도 비선형 형태를 취한다. 따라서, 디버깅에 투입되는 누적 인력 E_c는 식 (13) 으로 표현된다.

$$E_c = \alpha_2 m_c^2 + \beta_2 m_c + \gamma_2 \text{ or } E_c = \alpha_2 m_i^2 + \beta_2 m_i + \gamma_2 \quad (13)$$

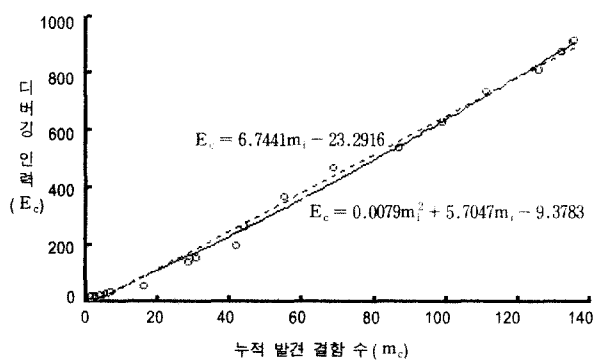
이들 제안된 모델의 모수들도 회귀분석을 거쳐 추정되었으며, 적합된 결과는 (그림 10)(a)와 (b)에 제시되어 있으며, 결과적으로 식 (14)와 식 (15) 모델을 얻었다.

$$E_c = -0.0187 m_c^2 + 9.2384 m_c - 0.7945 \quad (14)$$

$$E_c = 0.0079 m_i^2 + 5.7047 m_i - 9.3783 \quad (15)$$



(a) 누적 디버깅 결함 수에 따른 디버깅과정 투입인력 추정 모델



(b) 누적 발견 결함 수에 따른 디버깅과정 투입인력 추정 모델

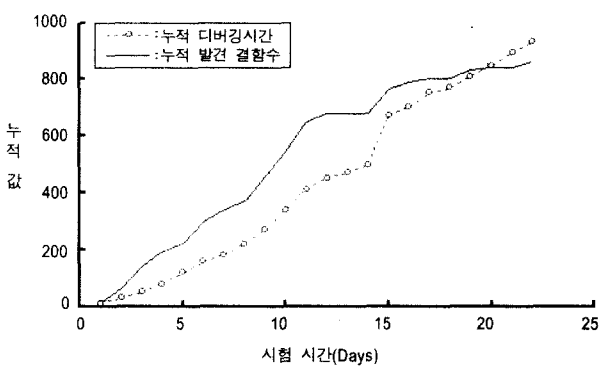
(그림 10) 누적 디버깅 과정 투입 인력 추정 모델

<표 4>에는 모델의 성능을 비교하고 있다. 디버깅 투입인력 추정 모델은 단지 18.38%와 23.46%의 오차만을 나타내며 실제 투입 인력을 추정할 수 있는 성능을 갖고 있음을 알 수 있다.

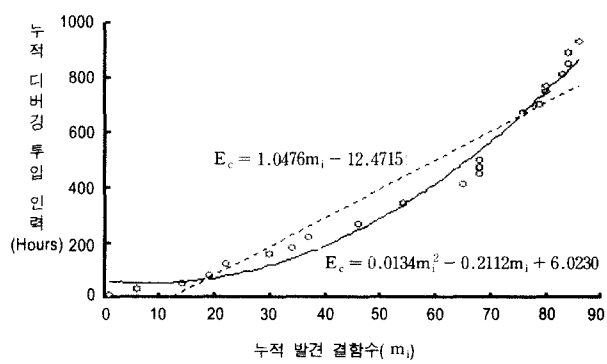
〈표 4〉 Data Set 4의 디버깅 인력 모델 성능 비교

인력 추정 모델		성 능	
		결정계수	MMRE
누적 디버깅 결함 수 기준	$E_c = -0.0187 m_c^2 + 9.2384 m_c - 0.7945$	0.9960	18.38%
	$E_c = 6.8545 m_c + 25.2380$	0.9889	70.10%
누적 발견 결함수 기준	$E_c = 0.0079 m_i^2 + 5.7047 m_i - 9.3783$	0.9958	23.46%
	$E_c = 6.7441 m_i - 23.2916$	0.9943	49.67%

추가적으로, Data Set 5에 대해 본 제안모델이 적합한지 평가하였다. Data Set 5는 Tohma, Jacoby, Murata와 Yamamoto[15]의 Table 3 데이터로 22일 동안 86개의 발견된 결함을 디버깅하는데 93시간이 투입되었다. (그림 11)과 같이 Data Set 5에 대해 시험시간에 따른 누적 결함 수와 누적 디버깅 인력 분포와 누적 결함 수와 누적 디버깅 투입 인력간 관계로부터 유도된 모델은 제안된 모델로 표현이 되며, <표 5>와 같이 98.05%의 설명력과 67.07%의 평균상대오차를 보이고 있어 본 제안 모델이 적합함을 알 수 있다.



(a) 시험시간에 따른 누적 디버깅 인력과 발견 결함 수



(b) 누적 발견 결함 수에 따른 누적 디버깅 투입인력

(그림 11) Data Set 5의 누적 디버깅 투입인력 추정 모델

〈표 5〉 Data Set 5의 디버깅 인력 모델 성능 비교

인력 추정 모델	성 능	
	결정계수	MMRE
$E_c = 0.0134m_i^2 - 0.2112m_i + 6.0230$	0.9805	67.07%
$E_c = 1.0476m_i - 12.4715$	0.9173	114.86%

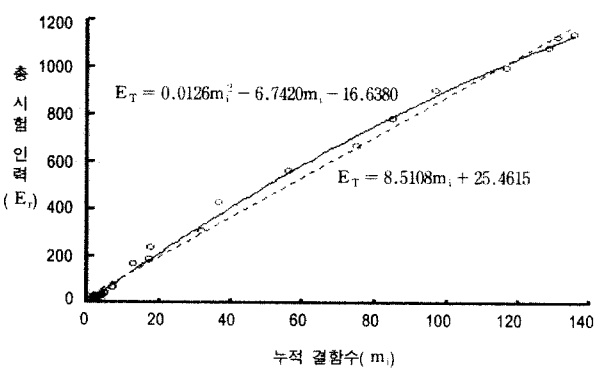
4. 시험단계 투입 인력 추정 모델

다음으로 시험단계에서 시험과 디버깅 과정 모두에 투입되는 총 인력을 추정하는 모델을 고려해 보자. Data Set 4에 대해 누적 결함 수 (m_i)와 총 시험인력 (E_T)과의 관계와 이들 관계로부터 유도된 모델은 (그림 12)에 제시되어 있다.

시험과 디버깅에 투입된 인력을 추정하는 식 (4)와 식 (5)의 모델과 동일하게 식 (15)의 곡선으로 표현되며, <표 6>에서 알 수 있듯이 제안된 비선형 모델이 좋은 추정 능력을 갖고 있음을 알 수 있다.

$$E_T = \alpha_3 m_i^2 + \beta_3 m_i + \gamma_3 \qquad (15)$$

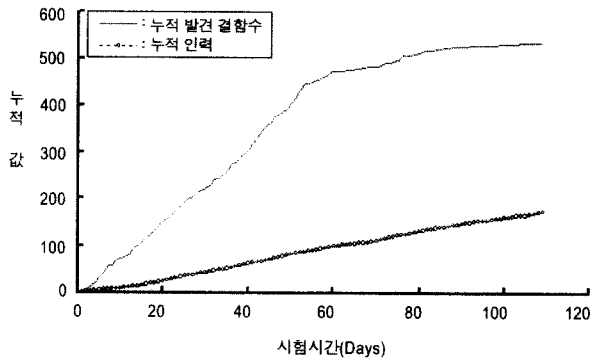
추가적으로 Data Set 2와 Data Set 3에 대해 제안된 모델이 적합한지 평가해보자. (그림 13)에서 알 수 있듯이 Data Set 2와 3 모두 발견된 누적 결함 수와 시험단계에서 투입된 누적 인력간의 관계가 곡선 형태를 갖고 있다. Data Set 2와 3에 제안된 모델을 적용한 결과는 (그림 13)에 제시되어 있다. 그림에서 알 수 있듯이 시험단계에서 발견되는 결점 수가 증가함으로 인해 투입되는 인력은 비선형적으로 증가하는 모델 형태를 취함을 알 수 있다.



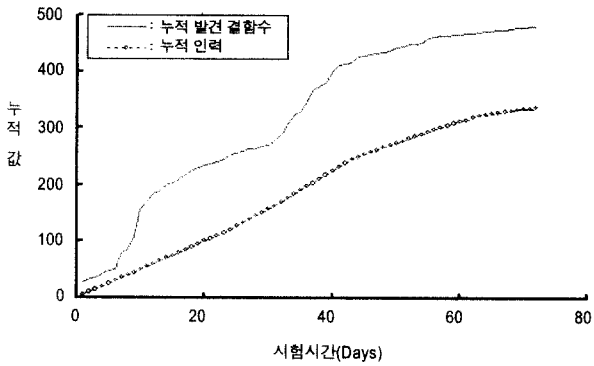
(그림 12) Data Set 4의 시험단계 투입인력 추정

〈표 6〉 Data Set 4의 총 시험인력 모델 성능 비교

인력 추정 모델	성 능	
	결정계수	MMRE
$E_T = 0.0126m_i^2 - 6.7420m_i - 16.6380$	0.9965	32.79%
$E_T = 8.5108m_i + 25.4615$	0.9924	79.57%



(a) Data Set 2



(b) Data Set 3

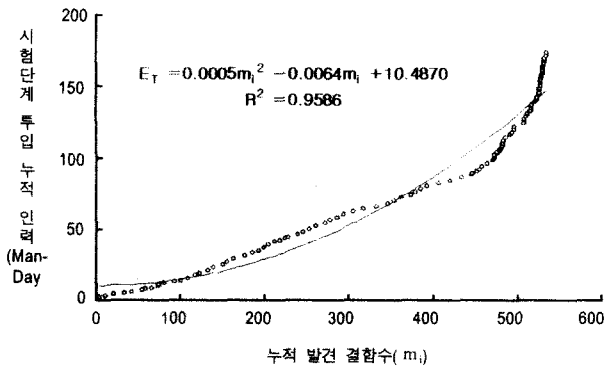
(그림 13) 누적 투입인력과 누적 결함 수(Data Set 2와 3)

결론적으로, 시험단계에서 시험, 디버깅 과정 각각에 투입되는 인력뿐만 아니라 이들 2개 과정을 모두 합한 총 투입인력도 발견되는 결함 수 또는 디버깅되는 결함 수와의 관계로부터 식 (16)의 다항식 곡선 형태의 모델로 표현이 가능함을 알 수 있다. 여기서, $a = T, i, c$ 이며, $b = i, c$이다.

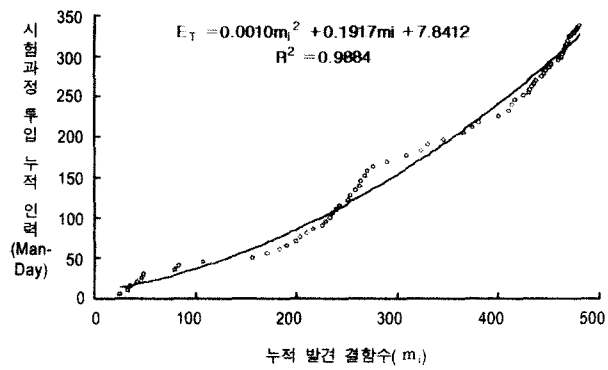
$$E_a = \alpha m_b^2 + \beta m_b + \gamma \qquad (16)$$

5. 결론 및 향후 연구과제

기존의 LOC나 FP를 이용한 개발인력 추정모델들은 소프트웨어 생명주기 전체에 소요되는 총 인력의 규모를 추정할 수 있다. 또한 Putman[8]의 SLIM 모델은 소프트웨어 생명주기 전체에 소요되는 인력의 분포뿐만 아니라 단위 시간에서 투입되는 인력의 규모도 추정할 수 있다. 그러나 이들 모델들은 생명주기의 세부단계인 요구사항 분석, 설계, 코딩, 시험과 유지보수 단계에 투입되는 인력의 규모와 변동을 추정할 수 없다. 특히, 시험단계는 다른 단계와 달리 시험과정과 디버깅과정이 수행되는 특징이 있다. 따라서, 시험단계에서 시험과정과 디버깅과정에 투입되는 인력의 규모를 얼마로 결정할 것인가가 문제로 발생한다.



(a) Data Set 2



(b) Data Set 3

(그림 14) 시험단계 투입 인력 추정 모델(Data Set 2와 3)

본 논문은 시험단계에 투입되는 인력의 규모뿐만 아니라 시험과정과 디버깅과정에 투입되는 인력의 규모도 추정할 수 있는 모델을 제안하였다. 제안된 모델의 특징은 Putnam[8]의 SLIM 모델과 같이 시간에 따른 인력 규모의 변동을 추정하지 않고, 시험단계에서 발견되는 결함 수 또는 디버깅되는 결함 수의 양에 따라 투입되는 인력의 규모를 추정할 수 있는 모델이다. 제안된 모델은 5개의 다른 소프트웨어 프로젝트들에 적용되어 적합성이 검증되었다.

제안된 모델은 시험단계에 대해 투입되는 인력의 규모만을 추정할 수 있다. 그러나 요구사항 분석, 설계, 코딩과 유지보수 단계에 대해서도 투입 인력의 규모를 추정하는 것이 성공적인 프로젝트 개발에 필수적인 관리요소이다. 따라서, 추후 이 분야에 대한 연구를 수행할 것이다.

참 고 문 헌

[1] K. H. Möller and D. J. Paulish, "Software Metrics - A Practitioners Guide to Improved Product Development," Chapman & Hall Co., New York, 1993.

[2] B. W. Boehm, "Software Engineering Economics," Prentice Hall, 1981.

[3] B. W. Boehm, "Software Engineering Economics," IEEE Trans. on Software Eng., Vol.10, No.1, pp.7-19, 1984.

[4] A. J. Albrecht and J. E. Gaffney, "Software Function, Source Line of Code and Development Effort Prediction : A Software Science Validation," IEEE Trans. on Software Eng., Vol.SE-9, No.6, pp.639-648, 1983.

[5] A. J. Albrecht, "Measuring Application Development Productivity," in Programming Productivity : Issues for the Eighties, C. Jones, ed. Washington, DC : IEEE Computer Society Press, 1981.

[6] C. F. Kemerer, "An Empirical Validation of Software Cost Estimation Models," Communication ACM, Vol.30, No.5, pp. 416-429, 1987.

[7] J. E. Matson, B. E. Barrett and J. M. Mellichamp, "Software Development Cost Estimation Using Function Points," IEEE Trans. on Software Eng., Vol.20, No.4, pp.275-287.

[8] L. H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," IEEE Trans. on Software Eng., Vol.SE-4, No.4, 1978.

[9] S. Yamada, H. Ohtera and H. Narihisa, "Software Reliability Growth Models with Testing-Effort," IEEE Trans. on Reliability, Vol.R-35, pp.19-23, 1986.

[10] S. Yamada, J. Hishitani and S. Osaki, "Software- Reliability Growth with a Weibull Test-Effort : A Model & Application," IEEE Trans. on Reliability, Vol.42, No.1, pp.100-106, 1993.

[11] K. Phillai and V. S. Sukumaran Nair, "A Model for Software Development Effort and Cost Estimation," IEEE Trans. on Software Eng., Vol.23, No.8, pp.485-497, 1997.

[12] P. V. Norden, "Project Life Cycle Modeling : Background and Application of the Life Cycle Curves," U. S. Army Computer System Command, 1977.

[13] P. V. Norden, "Curve Fitting for a Model of Applied Research and Development Scheduling," IBM J. Research and Development, Vol.3, No.2, pp.232-248, 1958.

[14] 이상운, "소프트웨어 시험노력 추정 시그모이드 모델", 정보처리학회논문지D, 제11-D권 제4호, pp.885-892, 2004.

[15] Y. Tohma, R. Jacoby, Y. Murata and M. Yamamoto, "Hyper-Geometric Distribution Model to Estimate the Number of Residual Software Faults," COMPSAC89, Orland, Florida, pp.610-617, 1989.

[16] T. Minohara and Y. Tohma, "Parameter Estimation of Hyper-Geometric Distribution Software Reliability Growth Model by Generic Algorithms," Proc. 6th Int'l Symp. Software Reliability Engineering, pp.324-329, 1995.

[17] J. D. Musa, A. Iannino and K. Okumoto, "Software Reliability Measurement, Prediction, Application," McGraw-Hill Book Company, 1987.

박 주 석

e-mail : iage2k@dreamwiz.com
1984년 해군사관학교 전자공학과(공학사)
1995년 국방대학원 전산계산학과(전산학석사)
2004년 숭실대학교 컴퓨터학과(공학박사)
2001년~현재 국방대학교 직무연수부 교수
관심분야 : 비용산정, 표준 및 프로세스, 정보체계 사업관리, 소프트웨어품질보증, 정보전