기능점수를 이용한 소프트웨어 개발노력 추정

이 상 운[†]·강 정 호[††]·박 중 양[†††]

요 약

소프트웨어공학에서 소프트웨어 측정분야는 30년 이상 수많은 연구가 있어 왔으나 아직까지 구체적인 소프트웨어 개발노력과 비용 추정 모델이 거의 없는 실정이다. 만약 소프트웨어 개발노력과 비용을 측정하려면 소프트웨어 규모를 추정해야 한다. 소프트웨어의 규모를 측정하기 위한 많은 소프트웨어 척도가 개발되었지만 가장 일반적인 척도가 LOC(line of code)와 FPA(Function Point Analysis)이다. FPA는 소프트웨어 규모를 측정하는데 LOC를 사용할 때의 단점을 극복할 수 있는 기법이다. 본 논문은 FP(Function Point)로 측정된 소프트웨어 규모로 소프트웨어 개발노력을 추정하는 단순회귀모델을 제안한다. 실험에 사용된 데이터들은 다양한 개발환경과 개발방법을 적용한 최근의 789개 소프트웨어 개발 프로젝트들이다. 실험 데이타들에 대한 산점도를 그려 개발노력과 FP의 적합한 관계로부터 단순회귀분석 모델을 유도하였다. 또한, 다양한 소프트웨어 개발환경과 개발방법 등을 고려해 개발된 최근의 대용량 프로젝트에 대해 제안된 단순회귀분석 모델이 기존의 회귀분석 모델들보다 개발노력 추정 정확성이 월등히 좋음을 보였다.

Software Development Effort Estimation Using Function Point

Sang-Un Lee[†]·Jeung-Ho Kang[††]·Joong-Yang Park[†††]

ABSTRACT

Area of software measurement in software engineering is active more than thirty years. There is a huge collection of researches but still no concrete software development effort and cost estimation model. If we want to measure the effort and cost of a software project, we need to estimate the size of the software. A number of software metrics are identified in the literature; the most frequently cited measures are LOC (line of code) and FPA (function point analysis). The FPA approach has features that overcome the major problems with using LOC as a measure of system size. This paper presents simple linear regression model that related software development effort to software size measured in FP. The model is derived from the plotting of the effort and FP relation. The experimental data are collected from 789 software development projects that were recently developed under the various development environments and development methods. Also, the model is compare with other regression analysis model. The presented model has the best estimation ability among the software effort estimation models.

키워드 : 소프트웨어 개발노력 추정(Software Effort Estimation), 소프트웨어 규모(Software Sizing), 회귀분석(Regression Analysis), 기능점수(Function Point), 변수 변환(Variable Transformation)

1. 서 론

소프트웨어 개발시 중요하게 제기되는 문제점으로 소프트웨어 생명주기의 초기단계에서 개발에 투입될 노력과 비용을 추정하는 능력이다. 소프트웨어 측정분야는 30년 이상 수많은 연구가 있어왔으나 소프트웨어 개발노력과 비용에 영향을 미치는 다양한 속성들과 이들간의 불명확한 관계로 아직까지 구체적인 소프트웨어 개발노력 및 비용측정 모델이 거의 없는 실정이다[1]. 실제로 소프트웨어 개발노력과 비용을 추정하려면 소프트웨어의 규모를 측정해야 한다. 길이로서 소프트웨어의 규모를 측정하기 위해 최초로 개발된 소프트웨어 척도 중 하나가 LOC(Line Of Code)이다[1, 2]. LOC 척도를 기초로 한 모델로는 Boehm[3, 4]의 COCOMO(Constructive Cost Model) 모델이 널리 사용되고 있다. 소프트웨어 규모 측정 단위로 LOC를 사용할 경우 LOC에 대한 일반적으로 받아들일 수 있는 정확한 정의 부족, 언어에 종속, 요구분석 또는 설계단계에서 정확한 LOC의 추정 어려움과 소프트웨어 규모를 측정함에 있어 특정한 하나의 관점인 길이만 고려하는 문제점이 있다[1].

이의 대안으로 프로젝트의 규모를 측정하는 척도로서 복잡도와 기능성에 대한 광범위한 연구가 이루어졌다. 복잡도를 측정하는 척도로는 Halsted's Software Science와 Mc-

※ 본 연구는 2001년도 경상대학교 연구년제 연구교수 연구지원비에 의하여 수행되었음.
† 정 회 원 : 국방품질관리소 항공전자장비 및 소프트웨어 품질보증 담당
†† 정 회 원 : 경남미래산업재단 정보화사업단 소프트웨어개발 팀장
††† 정 회 원 : 경상대학교 통계학과 교수
논문접수 : 2001년 7월 20일, 심사완료 : 2001년 10월 26일

Cabe's Cyclomatic Number가 있으며 개발 초기에 계산될 수도 있지만 대부분의 복잡도 척도가 LOC에 기초하고 있다. 기능성 척도로는 시스템의 기능성으로 소프트웨어 규모를 측정하는 FPA(Function Point Analysis)[5, 6]와 Demarco's Bang Metrics[7]가 있다. 과거 20년 이상 기능적 규모 척도가 소프트웨어공학 분야에서 연구되었으며, Bang Metrics에 대한 많은 연구를 수행하였으나 상업적으로 널리 사용되지 못하였는데 비해 FPA는 특정 분야에서 널리 사용되고 있다. FPA는 개발과정 초기에 프로그램의 기능적인 측면에서 소프트웨어 생산성을 측정하기 위한 척도로 IBM에 근무하는 Albrecht[5, 6]가 1979년에 제안하였다. FPA는 사용자에게 양도될 시스템의 기능에 기초하여 소프트웨어 시스템의 규모와 복잡도를 정량화하는 방법으로 소프트웨어 프로젝트를 개발하기 위해 사용되는 언어 또는 도구와 독립적이며, 개발 생명주기의 초기단계인 요구분석 단계에서 측정 가능한 장점이 있어 LOC를 사용할 때의 문제점을 극복할 수 있는 접근법이다[1].

기존의 FPA 기법을 이용해 소프트웨어 개발노력(Effort)을 추정하기 위한 모델들[1, 6-10]은 선형회귀 또는 다항식 회귀분석을 이용하여, FP에 따른 개발노력을 추정하였다. 그러나 이들 제안된 모델들이 사용한 표본의 크기가 작고, 대부분이 특정 업체에서 개발된 프로젝트를 대상으로 하고 있다. 또한 기존 모델들을 90년대에 다양한 개발환경 하에서 전세계 20여개 국에서 개발된 789개 프로젝트 데이터베이스를 갖고 있는 ISBSG(International Software Benchmarking Standards Group) Benchmark Release 6 데이터베이스[11]에 적용해본 결과 좋은 결과를 얻지 못하였다. 따라서, 본 논문에서는 FP를 이용해 표본의 크기가 상당히 크고, 다양한 개발환경 하에서 개발된 시스템들에 적합한 일반적인 소프트웨어 개발노력 추정 모델을 제안한다.

2장에서는 FPA 기법에 관한 관련연구 및 문제점을, 3장에서는 소프트웨어 개발노력 추정을 위해 일반적으로 적용 가능한 단순회귀 모델을 제시한다. 4장에서는 제안된 모델과 기존의 회귀분석 모델의 결과를 비교하여 제안된 모델이 가장 좋은 추정 결과를 얻음을 보인다.

2. 관련 연구 및 연구배경

2.1 LOC 이용 개발노력 추정 모델

LOC를 이용한 소프트웨어 비용 추정 모델은 식 (1) 또는 (2)의 형태를 취한다.

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

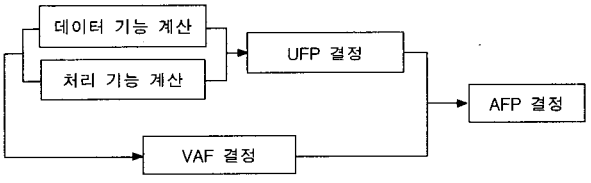
$$E = a \cdot (KLOC)^b \quad (2)$$

식 (1)과 식 (2)에서 E는 예측된 개발노력으로 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. 그러나 소프트웨어 규모 측정 단위로 LOC를 사용할 경우 발생되는 문제점은 다음과 같다[1].

- LOC에 대한 일반적으로 받아들일 수 있는 정확한 정의가 부족하다. Jones[12]는 라인을 계산하는 방법에 대해 11개의 변형된 방법을 확인했으며, 라인 계산 방법에 따라 약 500%의 불확실성을 가지고 있음을 지적했다.
- LOC는 언어에 종속되어 있다[13]. 다른 언어를 사용하여 개발된 프로젝트들에 대해 상호간에 직접 비교하기가 불가능하다. 예를 들면, 고차원 언어의 라인당 투입되는 시간은 저차원 언어보다 많다. 또한 동일한 기능을 제공하기 위해 고차원 언어에 대해 보다 적은 LOC가 요구됨은 자명한 사실이다.
- 요구분석 또는 설계단계에서 정확한 LOC 추정이 어려우며, 코딩이 종료된 후 측정이 가능하다[14]. 그러나, 소프트웨어 개발노력은 프로젝트 착수 초기에 보다 정확히 추정하여 사업관리 측면이나 자원 재분배 측면에서 활용할 수 있어야만 한다.
- 특정한 하나의 관점인 길이만을 고려하고 있으며, 소프트웨어의 기능성 또는 복잡도를 고려하지 못해 소프트웨어 특성을 정확히 반영하지 못한다.

2.2 FPA 기법

LOC를 이용한 개발노력 추정 모델의 단점을 보완하고자 소프트웨어의 복잡도와 기능성에 대한 광범위한 연구가 이루어졌으며, 시스템의 기능성으로 소프트웨어의 규모나 개발노력을 추정하는 FPA 기법이 개발되었다. 국제적으로 공인된 기능점수 FP를 계산하는 과정은 (그림 1)과 같다[15].



(그림 1) FP 계산과정

UFP(Unadjusted Function Point)는 프로젝트 또는 응용프로그램이 사용자에게 제공하는 특정한 계산 가능한 기능성(Functionality)을 나타낸다. 응용프로그램의 특정한 기능성은 응용프로그램에 의해 무엇이 사용자에게 양도될 것인가의 형태로 평가되며, 그것을 어떻게 양도되는가를 평가하

는 것이 아니다. 단지 사용자 요구사항과 정의된 컴퍼넌트(Components)가 계산된다. UFP는 데이터 기능(Data Function)과 처리 기능(Transaction Function)으로 구분되며, 데이터 기능은 다시 내부 논리적 파일(ILF, Internal Logical Files)과 외부 인터페이스 파일(EIF, External Interface File)로 세분화된다. 처리기능은 다시 외부 입력 (EI, External Inputs), 외부 출력(EO, External Outputs)과 외부 조회(EQ, External Inquiries)로 세분화된다.

데이터 기능은 내부와 외부 데이터 요구사항에 일치하도록 사용자에게 제공되는 기능성으로 내부 논리적 파일 또는 외부 인터페이스 파일중 하나이다. 내부 논리적 파일(ILF)는 사용자가 식별 가능한 응용프로그램의 범주(Boundary) 내에서 유지되는 논리적으로 관련된 데이터 또는 제어 정보 그룹이다. 외부 인터페이스 파일(EIF)는 사용자가 식별 가능한 응용프로그램에 의해 참조되는 논리적으로 관련된 데이터 또는 제어 정보의 그룹이며, 다른 응용프로그램의 범주 내에서 유지되지 않는다. 처리 기능은 처리 데이터에 대해 사용자에게 제공되는 기능성을 의미한다. 처리기능은 외부 입력, 외부 출력 또는 외부 조회중 하나이다. 외부 입력(EI)은 응용프로그램의 범주 외부로부터 오는 데이터 또는 제어 정보를 처리하는 기본적인 처리이다. 외부 출력(EO)은 응용프로그램의 범주 외부로 데이터 또는 제어 정보를 전송하는 기본적인 처리이다. 외부 조회(EQ)는 응용프로그램의 범주 외부로 데이터 또는 제어 정보를 보내는 기본적인 처리이다. *FP*는 다음 순서로 계산된다[15].

Step 1 : 5가지 기능요소 형태들(ILF, EIF, EI, EO 또는 EQ)의 수를 계산한다.

Step 2 : 기능요소 형태 각각에 대해 <표 1>의 기능적 복잡도 수준(Low, Average 또는 High)을 결정한다.

표에서 RET(Record Element Type)는 사용자가 인식할 수 있는 ILF 또는 EIF 내에 있는 데이터 요소의 서브그룹이다. DET(Data Element Type)는 사용자가 인지할 수 있는 유일하며, 반복이 없는 필드를 의미한다. 또한, FTR (File Types Referenced)는 처리기능으로 읽히거나 유지되는 ILF 또는 처리기능으로 읽히는 EIF를 의미한다.

Step 3 : 기능요소들을 i, 복잡도 수준을 j 라 하자. <표 1>에 의해 복잡도 수준 j에 있는 i번째 기능요소를 계산한 값을 z_{ij} 라하고 각 복잡도 수준에 대한 <표 2>의 가중치 w_{ij}와 선형결합 시키면 식 (3)의 UFP(Unadjusted FP)가 계산된다.

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} w_{ij} z_{ij} \qquad (3)$$

Step 4 : 사용자에게 제공되는 응용프로그램의 일반적인 처리 복잡도인 VAF(Value Adjustment Factor)를 계산한다.

〈표 1〉 복잡도 수준 계산

(a) ILF, EIF

구 분		DET		
		1~19	20~50	51 이상
RET	1	Low	Low	Average
	2~5	Low	Average	High
	6 이상	Average	High	High

(b) EI

구 분		DET		
		1~4	5~15	16 이상
FTR	0~1	Low	Low	Average
	2	Low	Average	High
	3 이상	Average	High	High

(c) EO, EQ

구 분		DET		
		1~5	6~19	20 이상
FTR	0~1	Low	Low	Average
	2~3	Low	Average	High
	4 이상	Average	High	High

〈표 2〉 가중치 w_{ij}

기능요소 형태	기능 복잡도		
	Low	Average	High
EI	× 3	× 4	× 6
EO	× 4	× 5	× 7
ILF	× 7	× 10	× 15
EIF	× 5	× 7	× 10
EQ	× 3	× 4	× 6

VAF는 응용프로그램의 일반적인 기능성을 평가하는 14개의 일반적인 시스템 속성(GSC, General System Characteristics)들로 구성되어 있다 :

① 데이터 통신(Data Communication),
② 분산 데이터 처리(Distributed Data Processing),
③ 성능(Performance),
④ 가중된 사용 형상(Heavily Used Configuration),
⑤ 처리율(Transaction Rate),
⑥ 온라인 데이터 입력(Online Data Entry),
⑦ 사용자 능률(End-User Efficiency),
⑧ 온라인 갱신(Online Update),
⑨ 복잡한 처리(Complex Processing),

⑩ 재사용성(Reusability),
⑪ 설치 용이성(Installation Ease),
⑫ 운영 용이성(Operational Ease),
⑬ 다중 설치운영(Multiple Sites),
⑭ 변경 용이성(Facilitate Change).

위 14개 GSC 각각에 대한 영향 정도(DI, Degree of Influence)는 다음과 같이 0~5의 범위로 평가한다 :

⓪ 존재하지 않거나 영향이 없음(Not Present or No Influence),
① 우발적 영향(Incidental Influence),
② 보통의 영향(Moderate Influence),
③ 평균적인 영향(Average Influence),
④ 중요한 영향(Significant Influence),
⑤ 완전히 강한 영향(Strong Influence Throughout).

14개 GSC 모두에 대한 DI를 합하여 TDI(Total Degree of Influence)를 계산한다. 즉, $TDI = \sum_{i=1}^{14} DI_i$. TDI를 식 (4)에 대입하여 VAF를 계산한다.

$$VAF = (TDI \cdot 0.01) + 0.65 \quad (4)$$

Step 5. 소프트웨어 프로젝트에 대한 최종 조절된 FP(AFP, Adjusted FP)는 UFP에 VAF를 곱하여 식 (5)로 계산된다.

$$AFP = VAF \cdot UFP = VAF \cdot \sum_{i=1}^{5} \sum_{j=1}^{3} w_{ij} z_{ij} \quad (5)$$

식 (5)로 얻은 AFP를 일반적으로 기능점수 FP라 칭한다.

2.3 FP 이용 소프트웨어 개발노력 추정 모델

FP 척도는 소프트웨어 프로젝트 추정, 생산성과 품질 평가에서 일관성과 객관성을 제공하는 구체적으로 공인되고 표준화된 척도로 프로젝트 계획 중에 소요 비용과 일정을 추정하는데 유용하게 사용되며, LOC에 비해 소프트웨어의 개발 초기 단계인 요구분석 단계에서 측정이 가능한 장점을 갖고 있다. FP를 이용한 개발노력 추정에 관한 연구는 대부분이 통계적인 회귀모델을 제시하였다. 일반적으로 적절한 회귀모델 선택방법은 다음 절차를 따른다[16].

Step 1 : 두 변수 x, y에 대한 자료 $(x_i, y_i), i=1,2,\cdots,n$을 수집하여 산점도를 그린다.
Step 2 : 산점도로부터 어떠한 회귀모델이 적절한지를 선택한다.
Step 3 : 선택된 회귀모델을 자료에 적합시켜 회귀 방정식을 구한 후, 이 회귀모델이 올바른 선택되었는지를 다음 기준에 의해 판단한다.
  a. 분산분석표에 의한 유의성 검정 결과 유의하지 않으면 선택된 회귀모델은 옳지 않다.
  b. 결정계수 R^2의 값이 충분히 크지 않으면 선택된 회귀모델이 옳지 않다.
  c. 잔차 분석을 통해 선택된 회귀모델이 옳은지를 판단한다.
Step 4 : Step 3의 판단기준에 의해 선택된 회귀모델이 옳지 않으면 Step 2로 복귀한다.

회귀분석의 경우 회귀직선에 의해 종속변수가 설명되는 정도를 결정계수(Coefficient of determination, R^2)라 한다[16]. 즉, 종속변수의 값은 독립변수에 의해 결정되는 부분과 미지의 오차의 합으로 나타나며, 총 변동을 설명하는데 있어서 회귀직선에 의해 설명되는 변동 비율이 $R^2 (0 \le R^2 \le 1)$이다. 따라서, R^2가 0에 가까우면 추정된 회귀직선은 쓸모가 없으며, 값이 클수록 쓸모 있는 회귀직선이 된다. 잔차(Residual)는 실제 값과 추정된 값과의 차이로, 잔차분석은 단순회귀모델에서 설정한 등분산성, 독립성, 정규성과 직선관계의 가정이 옳은가를 검토할 때 가장 많이 사용되는 방법이다. 따라서, 잔차가 어떤 일정한 형태를 취하지 않고 랜덤하게 분산되어 있는 경우가 좋은 모델이 된다.

Albrecht et al[8]은 IBM Data Processing Services에서 개발된 24개 응용 프로그램에 대해 개발노력 E가 FP에 대해 식 (6)의 선형형태를 취함을 연구하였고, Matson et al.[1]은 이들 데이터에 대해 개발노력 E가 FP에 대해 식 (7)의 비선형 형태를 취하는 것이 보다 정확한 추정을 할 수 있음을 보였다. 이들 데이터에 대해 이상운[17]은 신경망을 이용하여 개발노력 E를 추정하는 모델을 제시하였다.

$$E = -13.39 + 0.0545 FP \quad (6)$$
$$\sqrt{E} = 1.000 + 0.00468 FP \quad (7)$$

Kemerer[10]는 ABC 회사에서 개발된 15개의 소프트웨어 프로젝트에 대해 식 (8)~식 (10)의 선형과 다항식 모델을 제시하였다.

$$E = -121.57 + 0.3411 FP \quad (8)$$
$$E = -69.13 + 0.723 FP - 8.054 \times 10^{-4} FP^2 + 3.073 \times 10^{-7} FP^3 \quad (9)$$
$$E = 60.62 + 7.728 \times 10^{-8} FP^3 \quad (10)$$

그러나 이들 모델들은 표본의 개수가 작고 특정 업체에서 개발된 프로젝트에 대해 얻어진 모델로 다양한 프로젝트와 개발환경을 반영하지 못하고 있다. 따라서, Matson et al.[1]은 대형업체로부터 획득된 104개의 프로젝트를 이용해 모델을 유도하였다. 이들 업체들은 잘 훈련되고, 경험이 있는 시스템 개발 인력과 현존하는 소프트웨어 개발 방법론

과 도구를 사용하고 있으며, 또한 수년간 FPA를 사용하여 생산성과 품질을 측정한 경험을 갖고 있었다. 이들 프로젝트들은 다양한 응용 분야, 다양한 언어들로 구성되어 있다. Matson et al.[1]은 이와 같이 큰 표본을 사용해 식 (11)과 식 (12)의 모델을 제시하였다.

$$E = 585.7 + 15.12FP \tag{11}$$

$$\ln(E) = 2.51 + 1.00\ln(FP) \tag{12}$$

이들 연구결과를 종합한 결과와 모델의 성능은 <표 3>에 제시되어 있다.

〈표 3〉 FP를 이용한 개발노력 추정 모델 성능

표본수	모 델		모델성능 (R^2)
24개	Albrecht et al.	$E = -13.39 + 0.0545 \cdot FP$	87.4 %
	Matson et al.	$\sqrt{E} = 1.000 + 0.00468 \cdot FP$	89.9 %
15개	Kemerer	$E = -121.57 + 0.3411 \cdot FP$	55.3 %
		$E = -69.13 + 0.723FP - 8.054 \times 10^{-4}FP^2 + 3.073 \times 10^{-7}FP^3$	88.4 %
		$E = 60.62 + 7.728 \times 10^{-8}FP^3$	84.7 %
104개	Matson et al.	$E = 585.7 + 15.12FP$	65.2 %
		$\ln(E) = 2.51 + 1.00\ln(FP)$	53.4 %

표에서, 기존 제안된 모델들을 종합해보면 FP를 이용한 E 추정 모델, FP를 이용한 $\sqrt{E}$ 추정 모델과 자연로그함수 (Natural Logarithm Function) 형태인 $\ln(FP)$를 이용한 $\ln(E)$ 추정 모델임을 알 수 있다. 또한 특정 프로젝트뿐만 아니라 표본의 크기에 따라 적합한 모델이 다르며, 모든 프로젝트에 공통적으로 적용할 수 있는 일반적인 모델 (Universal Model)이 없음을 알 수 있다.

2.4 ISBSG 데이터베이스 이용 기존 모델 평가

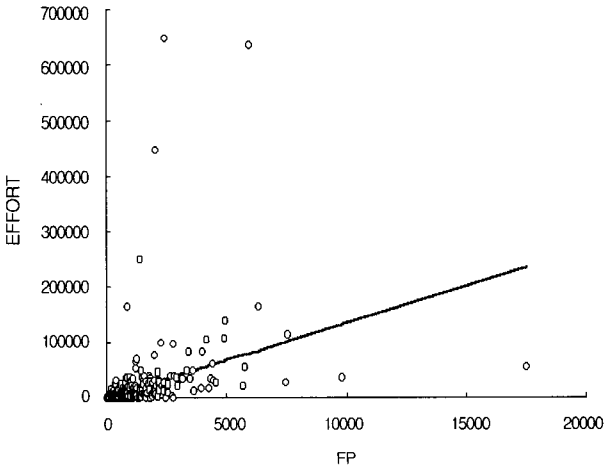
ISBSG Benchmark Release 6[11] 데이터베이스는 90년대에 20여개 나라에서 개발된 789개 프로젝트들에 대해 적용된 언어, 개발기법, 적용분야, 기능점수 FP, 개발노력 E와 개발기간 D등의 데이타를 갖고 있는 방대한 데이터베이스이다. <표 4>는 789개 프로젝트에서 표본으로 60개 프로젝트에 대한 기능점수 FP와 개발노력 E의 값을 보여주고 있다.

기존에 제안된 모델들을 평가하기 위해, 789개 프로젝트에 대해 기존 모델들인 개발노력 (E) vs. FP, $\sqrt{E}$ vs. FP와 $\ln(E)$ vs. $\ln(FP)$ 관계에대한 단순 회귀분석을 수행하였다. 단순회귀분석 수행 결과는 각각 (그림 2)~(그림 4)에 보여지고 있다. E vs. FP와 $\sqrt{E}$ vs. FP 관계의 경우, FP가 작은 프로젝트에서 큰 프로젝트 전반에 걸쳐 이상점들이 다수 발견되어 선형으로 적합이 불가능함을 알 수 있다. 또한 개발노력과 FP를 자연로그함수로 변환시킨 $\ln(E)$ vs. $\ln(FP)$ 관계는 선형관계가 아닌 타원형 형태를 취해 선형으로 적합시킨 모델의 성능이 좋지 않게 나올 수 있다. 각각의 관계에 대한 선형회귀모델과 성능은 <표 5>와 같다.

〈표 4〉 ISBSG Benchmark Release 6[11] 데이터 표본

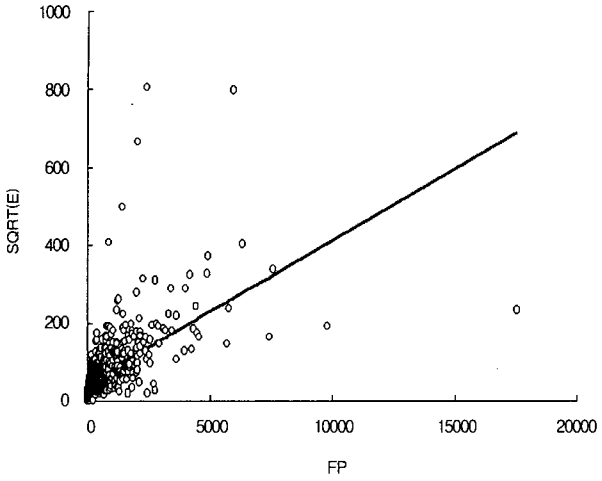
번호	프로젝트 ID	FP	Effort (Man-Month)	번호	프로젝트 ID	FP	Effort (Man-Month)
1	11929	9	239	31	11197	46	344
2	13304	10	60	32	10163	48	548
3	29971	11	97	33	15274	49	652
4	16665	12	211	34	18979	50	260
5	30773	12	10	35	30803	51	90
6	30567	13	1015	36	32243	51	338
7	16810	16	465	37	31627	53	849
8	15042	17	762	38	23232	54	67
9	24649	18	260	39	23271	54	1884
10	17737	25	140	40	23812	54	348
11	24586	25	580	41	11227	56	296
12	10500	29	880	42	27182	56	293
13	29775	30	880	43	31085	56	170
14	17065	31	237	44	32680	56	440
15	19209	32	352	45	14764	57	1179
16	31315	32	498	46	15411	57	1880
17	20860	33	145	47	18459	58	1040
18	32603	33	840	48	24659	58	1652
19	20619	38	571	49	11592	60	471
20	26686	38	247	50	29742	60	992
21	18873	39	242	51	16735	62	650
22	23802	39	114	52	14645	64	252
23	25965	39	21	53	20883	64	140
24	31978	39	1399	54	21311	64	1073
25	19488	40	285	55	28578	64	1544
26	10769	42	2496	56	21116	65	520
27	18245	43	331	57	16725	67	1584
28	17374	44	338	58	20971	68	910
29	28867	44	380	59	10816	69	129
30	12646	45	281	60	14371	70	432

<표 5>에서, FP와 E와의 3가지 관계 중 자연로그함수형태인 $\ln(E)$ vs. $\ln(FP)$ 관계를 이용한 선형회귀모델의 정확도가 52.73%로 가장 좋은 결과를 얻었다.

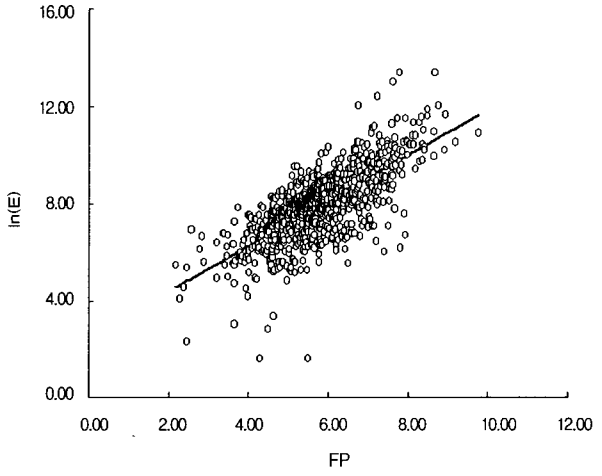


(그림 2) E 와 FP 관계 선형회귀

본 논문에서는 모델들의 성능을 비교 평가하기 위해 표준화된 잔차(Standardized Residual)분석과 더불어 상대오차 척도도 사용한다.



(그림 3) $\sqrt{E}$와 FP 관계 선형회귀

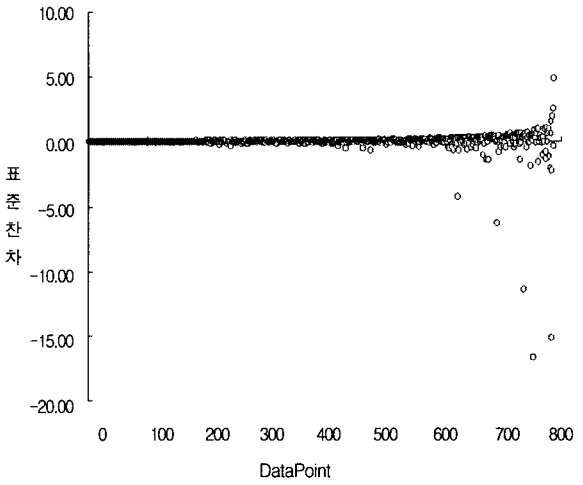


(그림 4) $\ln(E)$와 $\ln(FP)$와의 관계 선형회귀

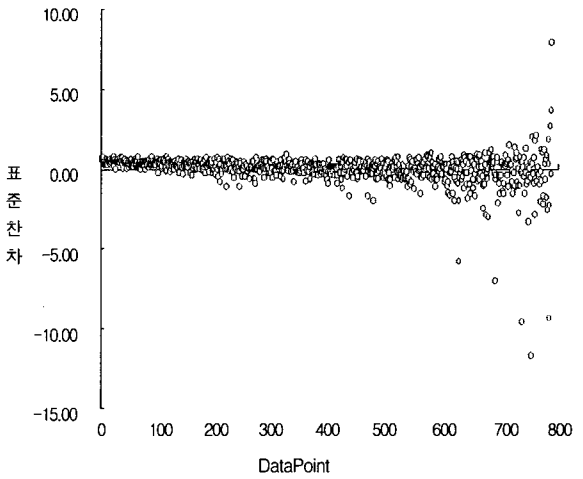
〈표 5〉 FP를 이용한 개발노력 추정 선형회귀모델

관 계	개발노력 추정모델	R^2
E vs. FP	$E=960.1676+13.3612FP$	14.81%
$\sqrt{E}$ vs.FP	$\sqrt{E}=45.5777+0.0365FP$	34.70%
$\ln(E)$ vs. $\ln(FP)$	$\ln(E)=2.4858+0.9351\ln(FP)$	52.73%

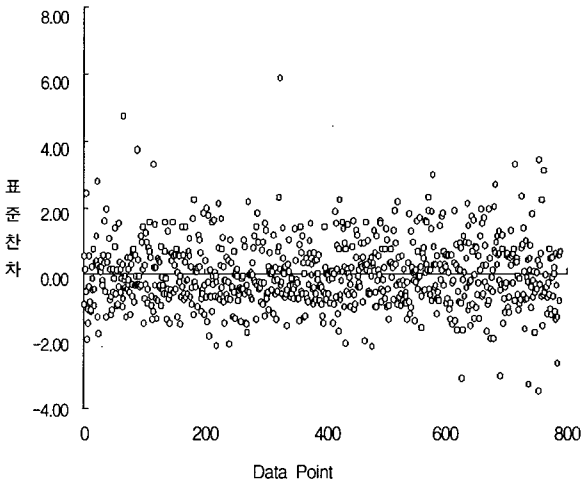
표준화된 잔차는 $\frac{\text{잔차}}{\text{잔차의 표준편차}}=\frac{\text{잔차}}{\sqrt{(\text{잔차}MSE)}}$로 계산된다. 위 3가지 모델에 대한 표준화된 잔차는 (그림 5)~(그림 7)에 나타내었다. 그림에서 보는바와 같이 E vs. FP와 $\sqrt{E}$ vs. FP 관계 모델은 잔차분석 결과 등분산성을 갖지 못하며, FP가 증가함에 따라 상당히 큰 값을 가짐을 알 수 있다. 다만 $\ln(FP)$를 이용한 $\ln(E)$ 추정모델이 잔차가 랜덤하게 골고루 분산되어 있어 적합한 모델임을 알 수 있으나 모델의 성능이 약 54%의 정확도를 가지는 단점이 있다.



(그림 5) FP 이용 E 추정 모델의 표준잔차



(그림 6) FP 이용 $\sqrt{E}$ 추정 모델의 표준잔차



(그림 7) $\ln(FP)$ 이용 $\ln(E)$ 추정모델의 표준잔차

따라서, 개발노력이 FP에 대해 비선형 관계를 가지는 모델이 보다 좋은 결과를 나타냄을 알 수 있다. 더군다나 FP의 크기에 관계없이 이상점(Outliers) 들이 많이 관측되므로, 선형 회귀분석만으로는 이들 데이터의 분포를 적절히 표현하지 못하며, 데이터에 대한 적절한 수학적인 변환이 필요함을 알 수 있다.

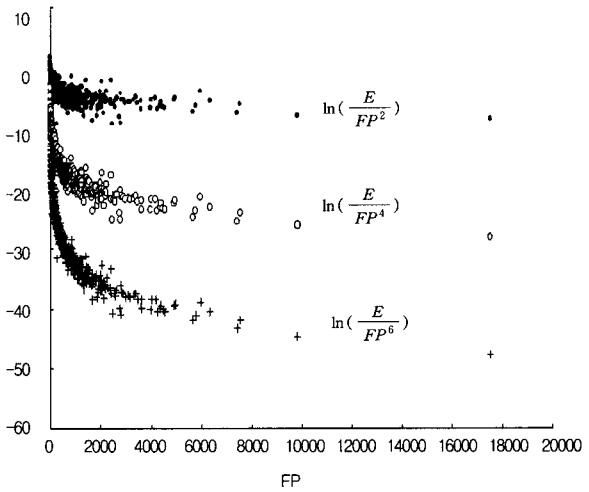
3. FP 이용 향상된 개발노력 추정 모델

개발노력 추정 분야의 대부분의 연구는 알고리즘 모델링에 초점을 두고 있다. 이 과정에서 개발노력은 척도를 가진 입력 또는 개발노력과 연관된 수학적 공식을 사용해 분석된다. 이 공식은 과거 이력자료 분석으로부터 제시된 형식적인 모델(Formal Model)을 사용하며, 모델의 정확도는 특정 개발환경을 반영한 가중치 조절을 수행하는 모델의 조정(Calibration) 과정을 통해 향상된다. 일반적으로 이 방법은 추정에 매우 큰 불일치성을 갖고 있다. Kemerer[10]는 예측된 값과 실측값 사이에 85~610%의 추정 오차가 있었으며, 모델의 조정을 거치면, 이들 모델의 성능을 향상시킬 수 있으나 역시 50~100%의 오차를 나타냄을 보였다. 따라서, 좋은 회귀모델 선택은 적정한 양의 데이터뿐만 아니라 주어진 모델의 신중하고 주의 깊은 분석을 요구한다.

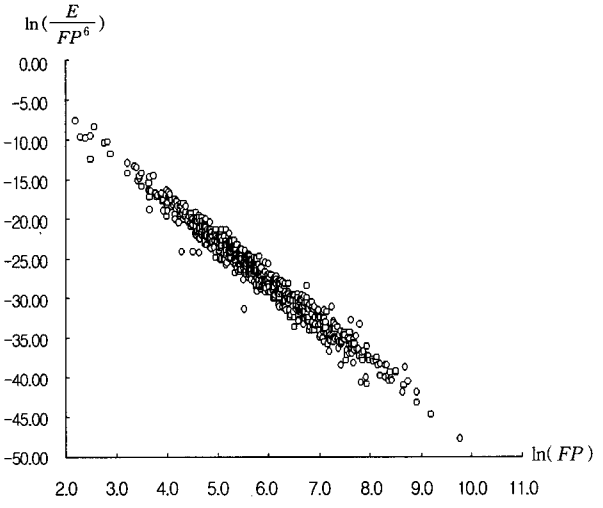
소프트웨어 개발노력 추정은 복잡한 문제이며, 기존 모델을 적용하기에는 다양한 소프트웨어 개발 특성을 고려하지 않았고, 적은 프로젝트 표본을 대상으로 개발되었기 때문에 실제로 적용하는데는 많은 제약을 갖고 있다. 따라서, 본 논문은 FP와 프로젝트 개발노력간의 관계에 대해 많은 표본과 다양한 환경에서 개발된 프로젝트를 대상으로 공통적으로 적용할 수 있는 일반화된 개발노력 추정 모델을 제시한다.

기존 모델들은 종속변수(E)를 설명하는데 독립변수(FP)의 일차함수로서 충분하다는 가정 즉, $E=a+b\cdot FP$라는 가정이 옳다고 생각하고 회귀분석을 수행하는 과정을 다루었다. 그러나 실제로 주변의 실제 상황들에 대해 산점도를 그려보면 직선보다는 다른 형태의 관계를 갖는 경우다 대부분이며, 자료의 분석에 있어서 단순회귀분석처럼 간단히 처리되지 않는다. 그러나 어떤 경우는 간단한 변수 변환(Variable Transformation)을 통해 직선관계로 변형시킬 수 있으며, 직선회귀 방법을 사용하여 분석한 후 변환되기 이전의 변수간의 관계를 규명하는 경우도 있다[16]. 본 논문은 이러한 방법을 이용하여 개발노력을 추정하는 일반화된 모델을 제시한다.

적절한 회귀모델을 찾기 위해, 주어진 데이터의 FP에 따른 개발노력 E와의 관계에 대한 산점도를 그려보아 적절한 관계를 도출해보자. FP값에 따른 $\ln(\frac{E}{FP^2})$, $\ln(\frac{E}{FP^4})$와 $\ln(\frac{E}{FP^6})$의 변화를 그려보면 (그림 8)과 같다. 그림에서 FP가 증가함에 따라 $\ln(\frac{E}{FP^2})$가 지수적으로 감소하는 관계는 갖고 있으나 산포된 결과를 나타낸다. 그러나 FP 증가에 따른 $\ln(\frac{E}{FP^6})$ 변화가 가장 명확하며, 지수적으로 감소하는 경향을 나타내고 있다. 따라서, 본 논문은 $\ln(\frac{E}{FP^6})$ vs. FP 관계를 선택하여 개발노력을 추정하는 모델을 제시한다. 그림에서 $\ln(\frac{E}{FP^6})$ vs. FP 관계 산점도로부터 회귀분석을 통해 모델을 유도할 수 있다. 그러나, 보다 단순한 모델을 얻기 위해, (그림 8)에서 가로축인 FP를 $\ln(FP)$로 변환시켜 $\ln(FP)$ 증가에 따른 $\ln(\frac{E}{FP^6})$의 변화를 그려보면 (그림 9)와 같이 선형적인 관계를 가짐을 알 수 있으며, 식 (13)의 모델에 대해 단순회귀분석을 수행하여 적합된 모델을 구할 수 있다.



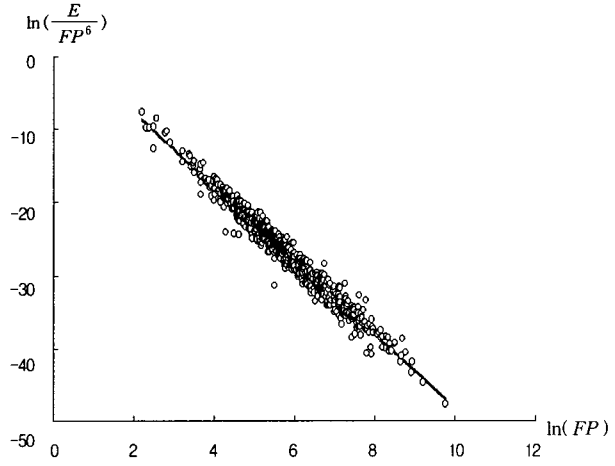
(그림 8) FP에 따른 E의 변환 관계



(그림 9) $\ln(\frac{E}{FP^6})$와 $\ln(FP)$ 관계

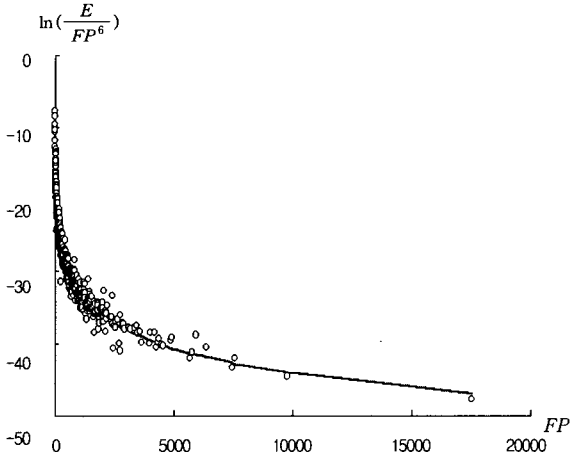
$$\ln(\frac{E}{FP^6}) = \alpha + \beta \cdot \ln(FP) \quad (13)$$

(그림 9)에 대해 회귀분석을 수행한 결과를 (그림 10)에 제시하였으며, 가로축의 값을 $\ln(FP)$ 대신 FP를 취하면 (그림 11)을 얻는다. 그림에서 산점도에 대해 회귀분석 결과 얻은 값을 적합시킨 결과 실제 데이터에 편차가 거의 없는 성능을 가진다. 회귀분석 결과 얻은 모델은 식 (14)에 제시되어 있다.



(그림 10) $\ln(\frac{E}{FP^6})$와 $\ln(FP)$ 관계 회귀분석

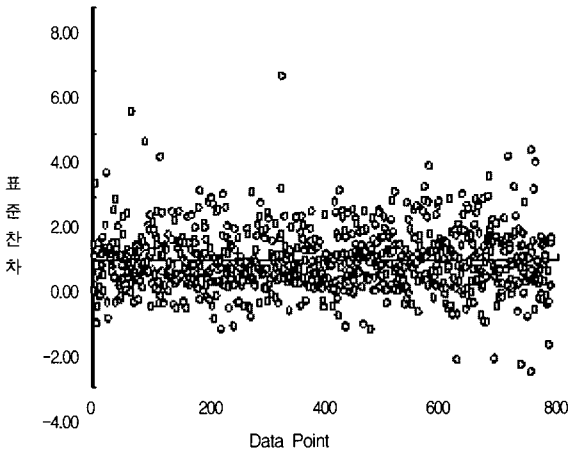
$$\ln(\frac{E}{FP^6}) = 2.4858 - 5.0649 \cdot \ln(FP) \quad (14)$$



(그림 11) $\ln(\frac{E}{FP^6})$와 FP 관계 회귀분석

4. 모델 평가

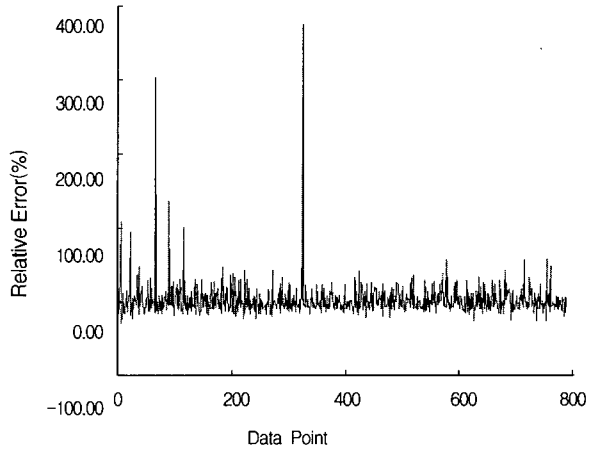
제안된 모델을 평가하기 위해 표준화된 잔차와 상대오차를 그려보자. 식 (14) 모델로 얻은 표준화된 잔차는 (그림 12)에 제시하였다. $\ln(FP)$ vs. $\ln(E)$ 관계 모델의 표준잔차인 (그림 7)과 비교시 유사한 결과를 얻어 두 모델간의 성능 차이가 분명하게 나타나지 않는다.



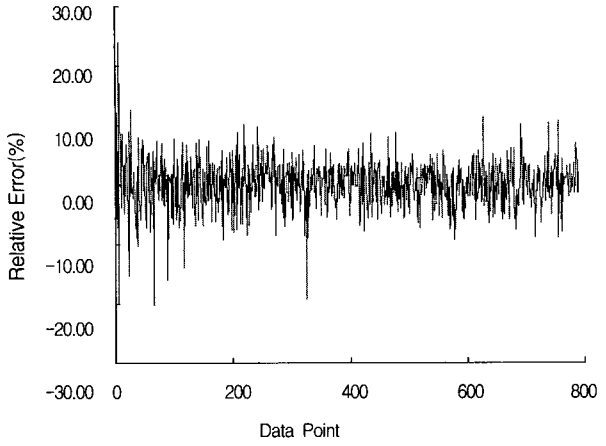
(그림 12) $\ln(\frac{E}{FP^6})$ vs. $\ln(FP)$ 관계 모델 표준잔차

따라서, 상대오차로 모델의 성능을 평가해 보자. 상대오차는 $\frac{(\text{추정치} - \text{실측치})}{\text{실측치}} \cdot 100\%$로 구해진다. $\ln(E)$ vs. $\ln(FP)$ 관계로부터 유도된 모델의 상대오차는 (그림 13), 제안된 모델인 $\ln(\frac{E}{FP^6})$ vs. $\ln(FP)$ 관계로부터 유도된 모델의 상대오차는 (그림 14)에 제시되어 있다. $\ln(FP)$ vs. $\ln(E)$ 관계 모델의 상대오차는 프로젝트 전반에 걸쳐 -20 ~+50%의 오차를, 몇몇 시스템에서는 최고 300~400%의 오차를 나타낸다. 반면에, 제안된 모델인 $\ln(\frac{E}{FP^6})$ vs. $\ln(FP)$ 관계로부터 유도된 모델은 전반적으로 ±10%의 오차를 보이며, 최대 -20%까지의 오차만을 보여 제안된 모델이 보다 좋은 추정 성능을 갖고 있다.

기존 제안된 모델들과 성능을 비교 분석한 결과는 <표 7>에 제시되어 있다. 제안된 모델의 성능이 97.04%로 실제 데이터의 약 3% 미만에 대해서만 설명하지 못하는 모델임



(그림 13) $\ln(FP)$ 이용 $\ln(E)$ 추정모델 상대오차



(그림 14) $\ln(FP)$ 이용 $\ln(\frac{E}{FP^6})$추정모델 상대오차

을 알 수 있으며, 기존 모델 중에 가장 성능이 좋은 $\ln(E)$ vs. $\ln(FP)$ 관계 모델의 성능인 52.73% 보다 약 45%의 성능향상을 보였다.

〈표 7〉 개발노력 추정 모델 성능 비교

관 계	개발노력 추정모델	R^2
E vs. FP	$E=960.1676+13.3612FP$	14.81%
$\sqrt{E}$ vs.FP	$\sqrt{E}=45.5777+0.0365FP$	34.70%
$\ln(E)$ vs. $\ln(FP)$	$\ln(E)=2.4858+0.9351\ln(FP)$	52.73%
$\ln(\frac{E}{FP^6})$ vs. $\ln(FP)$	$\ln(\frac{E}{FP^6})=2.4858-5.0649\ln(FP)$	97.04%

따라서, 제안된 모델이 기존 모델들 보다 월등히 좋은 개발노력 추정 능력을 갖고 있다. 개발될 프로젝트에 대한 FP값이 계산된 후 제안된 모델을 이용하여 소프트웨어 개발 초기에 개발에 투입될 노력을 추정한 결과가 프로젝트가 종료된 후 실제 투입된 개발노력과 비교시 3% 이내의 오차만을 가짐을 알 수 있다. 따라서, 프로젝트 관리측면에서 본 모델을 적용시 다양한 의사결정, 소요 예산 및 개발인원 할당과 계약체결 여부에 신뢰할 만한 정보를 얻을 수 있다.

5. 결론 및 향후 연구과제

FP를 이용해 개발노력을 추정하는 기존의 모델들은 사용된 표본의 개수가 적고, 개발환경과 개발방법 등 다양한 형태의 프로젝트에 적용되지 않는 문제점이 있다. 이의 증거로, ISBSG Benchmark Release 6[11]의 최근 다양한 개발환경과 방법론으로 개발된 789개 프로젝트에 적용한 결과 모델의 성능이 현저히 저하된 결과를 얻었다. 따라서, 본 논문은 표본의 크기가 상당히 크고, 다양한 최근이 소프트웨어 개발 환경과 방법론을 적용한 프로젝트에 적합한 소프트웨어 개발노력을 추정하는 모델을 제안하였다.

모델을 유도하기 위해, 실험 데이터에 대해 FP와 개발노력 E간의 관계가 명확한 산점도를 그리고, 이를 이용하여 단순회귀분석을 통해 모델을 제안하였다. 다양한 소프트웨어 개발환경과 개발방법 등을 고려해 개발된 최근의 대용량 프로젝트에 대해 제안된 단순회귀분석 모델의 성능이 기존의 가장 성능이 좋은 회귀분석 모델보다 45% 향상된 97.04%의 정확성을 가지고 있다. 따라서, 최근 개발되는 소프트웨어 프로젝트의 FP만 계산되면, 본 모델을 이용해 최적의 소프트웨어 개발노력을 추정하여 사업관리와 개발인력의 배분에 적용이 가능하다.

본 논문은 소프트웨어의 다양한 개발 환경과 개발방법 등을 상세히 고려하지 않고 단지 획득된 FP에 대한 소프트웨어의 개발노력을 추정하는 모델을 제시하였다. 그러나, 개발환경의 변화, 개발방법론의 차이로 인해 FP에 따른 개발노력도 변동될 수 있다. 따라서, 추후 이 분야에 대한 연구를 수행할 것이다.

참 고 문 헌

[1] 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, 1994.

[2] L. A. Laranjeira, "Software Size Estimation of Object-Oriented Systems," IEEE Trans. Software Eng., Vol.16, pp. 64-71, 1990.

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

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

[5] A. J. Albrecht, "Measuring Applications Development Productivity," Proceedings of IBM Application Dev., Joint SHARE/GUIDE Symposium, Monterey, CA., pp.83-92, 1979.

[6] 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.

[7] T. Demarco, "Controlling Software Projects Management Measurement & Estimation," New York : Yourdon Press, 1982.

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

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

[10] C. F. Keremer, "Reliability of Functional Point Measurement-A Field Experiment," Communications of ACM, 1993.

[11] ISBSG, "Worldwide Software Development - The Benchmark Release 6," Victoria, Australia International Software Benchmarking Standards Group, 2000.

[12] C. Jones, "Programming Productivity," New York, McGraw-Hill, 1986.

[13] G. C. Low and D. R. Jeffery, "Function Points in the Estimation and Evaluation of the Software Process," IEEE

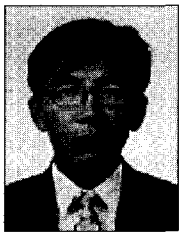
Trans. on Software Eng., Vol.16, pp.64-71, 1990.
[14] R. D. Emrick, "In Search of a Better Metric for Measuring Productivity of Application Development," Int. Function Point Users Group Conf. Proc., 1987.
[15] Bradley, M., "Function Point Counting Practices Manual, Release 4.1," International Function Point Users Group (IFPUG), 1999.
[16] 김우철, et al., "현대 통계학", 영지출판사, 1994.
[17] 이상운, "신경망을 이용한 소프트웨어 개발노력 추정", 정보처리학회논문지 D, 제8-D권 제3호, pp.241-246, 2001.

이 상 운

e-mail : sangun_lee@hanmail.net
1983년~1987년 한국항공대학교 항공전자공학과(학사)
1995년~1997년 경상대학교 컴퓨터과학과(석사)
1998년~2001년 경상대학교 컴퓨터과학과(박사)
1992년~현재 국방품질관리소 항공전자장비 및 소프트웨어 품질보증 담당
관심분야 : 소프트웨어 공학(소프트웨어 시험 및 품질보증, 소프트웨어 신뢰성), 소프트웨어 메트릭스, 신경망, 뉴로-퍼지

강 정 호

e-mail : kjh630107@hanmail.net
1990년 경상대학교 전산통계학과 학사
2001년 경상대학교 정보공학 석사
현재 경남미래산업재단 정보화사업단 소프트웨어개발 팀장
관심분야 : 소프트웨어공학, 전자상거래, 웹프로그래밍



박 중 양

e-mail : parkjy@nongae.gsnu.ac.kr
1982년 연세대학교 상경대학 응용통계학과 (학사)
1984년 한국과학기술원 산업공학과 응용통계전공(석사)
1994년 한국과학기술원 산업공학과 응용통계전공(박사)
1984년~1989년 경상대학교 전산통계학과 교수
1989년~현재 경상대학교 통계학과 교수
관심분야 : 소프트웨어 신뢰성, 선형 통계 모형, 신경망