論 文
5-4-7

소프트웨어 규모 측정 방법 연구
(A Study for Software Sizing Method)

박석규 (Seok-Gyu Park)[1] 박중양(Joong-Yang Park)[2]

요 약

소프트웨어 개발노력, 기간과 비용을 추정 능력은 개발될 소프트웨어의 정확한 규모 추정치에 기반한다. 본 논문은 소프트웨어 규모 추정을 위한 단순화된 기능점수 (FP) 기법을 제안한다. 이 기법은 가치조절인자를 계산하는 단계를 생략하고 조절인 안된 기능점수에서 조절된 기능점수를 직접 구한다. 783개의 소프트웨어 프로젝트에 기반을 둔 사례분석으로 통계적 회귀분석을 통해 적절한 모델을 찾고자 하였다. 본 논문은 또한, 신규 개발, 유지보수와 재개발 형태의 프로젝트 서브그룹별로 모델을 제시하였다.

ABSTRACT

A estimating capability of software effort, duration and cost is based on accurate size estimate of the software to be developed. A simplified function point (FP) approach to software size estimation is described, which first skip the computation step for value adjustment factor, thus directly obtaining final adjusted FP from unadjusted FP. The research seeks suitable models based on statistical regression models in the context of case study based on 783 software projects. The approach also are built for subsets of projects using new development, enhancement and re-development types.

1) 정회원 : 경상대학교 대학원 컴퓨터과학과
박사과정
2) 정회원 : 경상대학교 통계정보학과 교수

논문접수 : 2004. 4. 7.
심사완료 : 2004. 4. 14.

1. 서 론

소프트웨어 개발시 중요하게 제기되는 문제점으로 소프트웨어 생명주기의 초기단계에서 개발에 투입될 노력과 비용을 추정하는 능력이다[1,2]. 실제로 소프트웨어 개발노력과 비용을 추정하기 위해서는 먼저 규모를 측정해야 한다. 소프트웨어의 규모를 측정하는 대표적인 측도로는 라인 수 (Line Of Codes, LOC)와 기능점수 (Function Point, FP)가 있다[2]. LOC는 프로그램 언어에 따라 변하며, 소프트웨어 개발 초기인 요구분석 단계에서 정확한 추정이 어렵다. IBM 회사는 70년대 말 프로그램 언어에 따라 변동되지 않는 소프트웨어 규모 측도를 개발하여 소프트웨어 개발에 소요되는 노력과 비용을 추정하고자 하였다. 이에 따라 Albrecht[3]가 1979년에 기능점수 분석법 (Function Point Analysis, FPA)을 제안하였다. FPA는 사용자에게 제공될 시스템의 기능에 기반을 두고 소프트웨어 시스템의 규모와 복잡도를 정량화하는 방법으로 소프트웨어 프로젝트를 개발하기 위해 사용되는 언어, 적용도구와 독립적이며, 개발 초기인 요구분석 단계에서 측정 가능한 장점이 있어 LOC를 사용할 때의 문제점을 극복할 수 있는 접근법이다[2]. FPA 기법은 Albrecht[3,4]에 의해 제안된 이래, 20여년 동안 많은 연구가 수행되었다. 대표적인 사례가 국제적으로 공인된 ISBSG (International Software Benchmarking Standards Group) [5]가 있으며, 20여개 국에서 기능점수에 기반하여 1989년부터 2000년까지 개발된 789개 프로젝트에 대한 데이터베이스를 갖고 있다]. 기능점수는 5개의 소프트웨어 기능요소 (Function Element)들인 입력, 출력, 조회, 파일과 인터페이스의 수를 식별하고, 각각의 기능요소들에 대한 복잡도 (Low, Medium, High)에 대한 가중치 (Weight)를 곱하여 조절이 안된 기능점수 (Unadjusted FP, UFP)가 계산된다. 여기에 소프트웨어 프로젝트 수준의 시스템 복잡도인 가치조절인자 (Value Adjustment Factor, VAF)를 곱하여 조절된 기능점수 (Adjusted FP, AFP)가 계산된다. 프로젝트 수준의 복잡도 평가는 기능요소를 계산하는 실무자들이 감당하기 어려운 문제로 일반적으로 시스템 전문가가 측정한다[6]. 이 시스템 수준의 복잡도 값에 따라 UFP가 AFP로 변환되면서 최대 ±35%의 편차를 보인다[2]. 이와 같이 측정된 소프트웨어 규모인 조절된 기능점수가 정확해야만 이에 기반을 두고 추정되는 개발노력, 비용과 일정의 결과도 정확해질 수 있다. 따라서, 시스템 수준의 복잡도를 정확히 평가하는 것은 대단히 어려운 문제이면서도 중요한 요소라 할 수 있다.

본 논문은 최근 10여년 동안 개발된 다양한 프로젝트들의 이력자료에 기반하여 프로젝트 수준의 시스템 복잡도인 VAF를 계산하는 과정을 생략하고 조절 안된 기능점수로부터 조절된 기능점수를 직접 추정할 수 있는 모델을 제시한다. 본 제안 모델을 활용시, 기능점수 계산의 단순성을 도모함과 동시에 시스템 전문가의 도움 없이 기능점수 측정 과정의 정확성도 유지하는 장점을 갖고 있다.

2장에서는 FPA 기법에 관한 관련연구를, 3장에서는 조절인 안된 기능점수로부터 조절된 기능점수로 계산이 가능한 단순회귀 모델을 제시하고 평가해 본다.

2. 기능점수

조절이 안된 기능점수 UFP는 프로젝트 또는 응용프로그램이 사용자에게 제공하는 계산 가능한 기능성 (Functionality)을 의미하며, 데이터 기능 (Data Function)과 처리 기능 (Transaction Function)으로 구분된다. 데이터 기능은 내부와 외부 데이터 요구사항에 일치하도록 사용자에게 제공되는 것으로, 화일 (Internal Logical Files, ILF)과 인터페이스 (External Interface File, EIF)로 분류된다. 처리 기능은 처리된 데이터를 사용자에게 제공하는 것으로 입력 (External Inputs, EI), 출력 (External Outputs, EO)과 조회 (External

Inquiries, EQ)의 3가지 기능 요소 (Function Element)로 세분화된다[6].

소프트웨어 프로젝트 각각의 모듈 또는 컴포넌트에 대해 다음의 단계 3까지 계산하여 얻어진 결과가 UFP가 된다.

단계 1.

5가지 기능요소 형태들 (ILF, EIF, EI, EO 또는 EQ)을 분별 한다.

단계 2.

기능요소 형태 각각에 대해 <표 1>의 기준에 근거하여 복잡도 수준 (Low, Average 또는 High)을 결정한다.

<표 1> 복잡도 수준

<Table 1> Complexity Level

(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

표에서 RET (Record Element Type)는 사용자가 인식할 수 있는 파일 또는 인터페이스에 있는 데이터 요소이다. DET (Data Element Type)는 사용자가 인지할 수 있는 유일하며, 반복이 없는 필드를 의미한다. 또한, FTR (File Types Referenced)는 처리기능으로 읽히거나 유지되는 화일 또는 처리되는 인터페이스를 의미한다.

단계 3.

기능요소들을 $i(i=1,2,\cdots,5)$, 복잡도 수준을 $j(j=1,2,3)$ 라 하자. 복잡도 수준 j에 있는 i번째 기능요소를 z_{ij} 라하고 각 복잡도 수준에 대한 <표 2>의 가중치 w_{ij}와 선형 결합시키면 식 (2)와 같이 UFP가 계산된다.

$$UFP = \sum_{i=1}^{5}\sum_{j=1}^{3} w_{ij} z_{ij} \qquad (2)$$

단계 4.

사용자에게 제공되는 시스템 수준의 일반적인 처리 복잡도인 가치 조절 인자 VAF를 계산한다.

<표 2> 가중치 w_{ij}

<Table 2> Weights w_{ij}

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

VAF는 응용프로그램의 일반적인 기능성을 평가하는 14개의 일반적인 시스템 속성 (General System Characteristics, GSC)들로 구성되어 있다: (1) 데이터 통신; (2) 분산 데이터 처리; (3) 성능; (4) 가중된 사용 형상; (5) 처리율; (6) 온라인 데이터 입력; (7) 사용자 능률; (8) 온라인 갱신; (9) 처리 복잡도; (10) 재사용성; (11) 설치 용이성; (12) 운영 용이성; (13) 다중 설치운영; (14) 변경 용이성.

위 14개 GSC 각각에 대한 영향 정도 (Degree of Influence, DI)를 다음과 같이 0 ~ 5의 범위로 평가한다: (0) 관련 없거나 영향 없음; (1) 우발적 영향; (2) 보통의 영향; (3) 평균적 영

향; (4) 중대한 영향; (5) 강력한 영향.

14개 GSC 모두에 대한 DI를 합하여 TDI (Total Degree of Influence)를 계산한다. 즉, $TDI = \sum_{i=1}^{14} DI_i$. 식 (3)으로 VAF가 계산된다.

$$VAF = 0.65 + \frac{TDI}{100}, (0.65 \le VAF \le 1.35) \quad (3)$$

단계 5.

소프트웨어 프로젝트에 대한 최종 조절된 기능점수 AFP는 UFP에 VAF를 곱하여 식 (4)로 계산된다.

$$AFP = VAF \cdot UFP = VAF \cdot \sum_{i=1}^{5} \sum_{j=1}^{3} w_{ij} z_{ij} \quad (4)$$

식 (4)로 얻은 AFP를 일반적으로 기능점수 FP라 부른다. 비록 기능점수 기법의 기본원리는 단순하고 간단하지만 이들 원리를 수 천 종류의 다양한 소프트웨어 프로젝트에 실제 적용하는 것은 결코 단순하지 않다. 다른 개인들에 의해 계산된 FP의 차이가 150% (LOC의 경우와 동일) 이상 변동이 발생하면 유용한 측도 (Metrics)로 고려할 가치가 없다[7].

IFPUG (International Function Point User's Group) 지침서[6]는 VAF 계산 단계를 기능점수 계산과정의 마지막 부분에서 언급하고 있다. 이와 같이 하는 이유는 기능점수를 올바르게 계산하기 위해서는 VAF가 반드시 필요하며, 기능점수를 계산하는 실무자는 VAF에 대한 평가의 어려움으로 계산을 수행하는 시점을 계속 지연시키려고 하기 때문이다. 또 다른 이유는 시스템 전문가가 시스템의 특정한 특징만으로는 기능점수를 계산하는 것이 충분하지 않음을 제기하는 일이 종종 발생하기 때문이다. 기능점수를 계산하는 실무자는 이 특징을 VAF가 계산될 때 반영할 것을 요구하고 있다. VAF는 전체적인 시스템의 복잡도를 파악해 주는 것으로 시스템 전문가의 도움을 받아 보다 정확한 계산을 할 수 있다[5]. 또한, VAF의 값에 따라 실제 계산된 AFP는 UFP에 대해 ±35%의 편차를 가진다. 따라서, VAF를 잘못 측정하였을 경우 최종적으로 계산된 소프트웨어 규모에 심한 오차를 발생시킬 수 있다.

3. 단순화된 기능점수 계산 모델

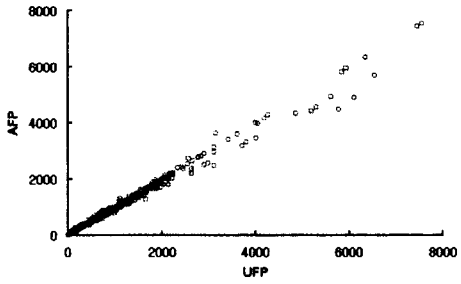
기능점수 계산을 보다 쉽게 하기 위해 VAF를 계산하는 과정인 단계 4를 제거하고 단계 3에서 구해진 UFP로부터 직접 단계 5의 AFP를 계산할 수 있는 모델을 제시한다. 제시되는 모델의 성능은 결정계수와 평균 상대오차로 평가한다. 이들 평가기준을 살펴보자. 회귀분석 결과 제안된 모델이 좋은지 여부를 평가하는 기준들 중 하나로 결정계수 (Coefficient of determination, R^2)를 계산한다. 회귀모델의 경우, 종속변수의 총 변동 (Total Sum of Squares, SST) 값은 독립변수에 의해 결정되는 부분 (Regression Sum of Squares, SSR)과 미지의 오차의 합 (Residual Sum of Squares, SSE)으로 나타난다. 총 변동 중에서 회귀직선으로 설명할 수 있는 비율을 결정계수 ($R^2 = \frac{SSR}{SST}$)라 한다. 따라서, $R^2 (0 \le R^2 \le 1)$이 값이 클수록 쓸모 있는 회귀직선이 된다[8]. 상대 오차 (Relative Error, RE)는

$$\frac{(\text{실측치} - \text{추정치})}{\text{실측치}} \times 100(\%)$$이다.

MRE (Magnitude of the RE)는 $|RE|$이며, n개의 데이터에 대한 $MMRE$ (Mean MRE) $= \frac{1}{n} \sum_{i=1}^{n} MRE_i$로 계산된다. 따라서, $MMRE$가 작은 값이면 모델은 평균적으로 좋은 모델이 된다[9]. Conte et al.[9]는 $MMRE \le 0.25$ (25%) 이면 적절한 모델로 채택 가능한 것으로 고려하였다.

3.1 사례연구 I

실험에 적용된 데이터는 ISBSG Benchmark Release 6[10]의 789개 프로젝트들이다. UFP로부터 직접 AFP를 얻기 위한 모델을 유도하기 위해 먼저, UFP와 AFP 관계에 대해 [그림 1]의 산점도를 그렸다.



[그림 1] UFP와 AFP 관계
[Fig. 1] *UFP* versus *AFP*

[그림 1]의 산점도로부터 UFP와 AFP가 선형 관계를 갖고 있음을 알 수 있다. 그러나 실제로는 비선형 으로 관계를 보다 적절히 표현할 수 있다. 이 경우 간단한 변수 변환 (Variable Transformation)을 통해 직선관계로 변형시켜 직선회귀 방법을 사용하여 분석한 후 변환되기 이전의 변수간의 관계를 규명하는 경우도 있다 [8]. 따라서, [그림 1]의 관계에 대해 선형회귀와 [그림 2]와 같이 로그 변환을 통한 비선형 관계 회귀분석을 수행한 결과 식 (5)와 (6)의 모델을 얻었다. 두 모델에 대한 성능 분석 결과는 <표 3>에, 상대오차는 [그림 3]에 제시되어 있다.

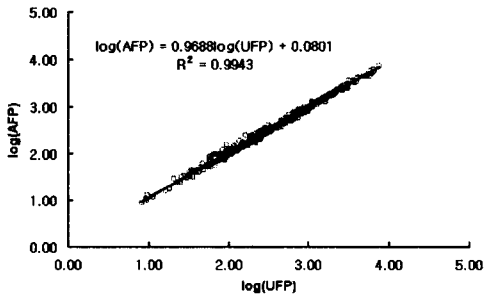
$$AFP = 0.9370\,UFP + 21.3255 \qquad (5)$$

$$AFP = 1.2026\,UFP^{0.9688} \qquad (6)$$

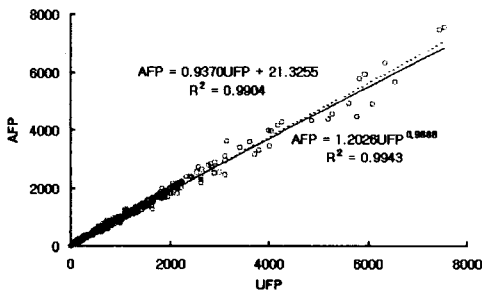
<표 3> 선형과 비선형 모델 성능
<Table 3> Performance of Linear and Nonlinear Model

모델	성능	
	결정계수	MMRE
$AFP = 0.9370\,UFP + 21.3255$	0.9943	10.75%
$AFP = 1.2026\,UFP^{0.9688}$	0.9904	6.04%

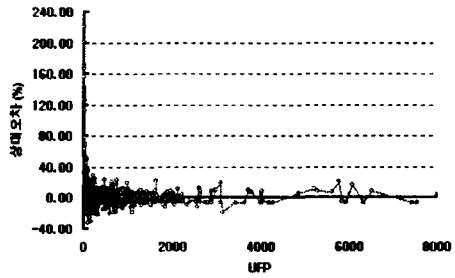
[그림 3]에서 알 수 있듯이 상대오차 측면에서 볼 때 2개 모델간에는 거의 유사한 성능을 보이나 규모가 작은 프로젝트에서는 심한 차이점을 갖고 있다. 따라서, 기능점수가 100이하인 규모의 프로젝트들을 제거하고 상대오차를 (Fig. 4)에 제시하였다.
100 이하의 기능점수를 가진 프로젝트들을 제거시 두 모델 모두 거의 동일한 상대오차를 보임을 알 수 있다. 따라서, 측정된 UFP가 100 이상일 경우 어느 모델을 사용해도 무방하지만, 100 이하의 UFP인 경우에는 $AFP = 1.2026\,UFP^{0.9688}$ 모델을 이용해 AFP를 구하는 것이 보다 적절한 AFP를 얻을 수 있다.



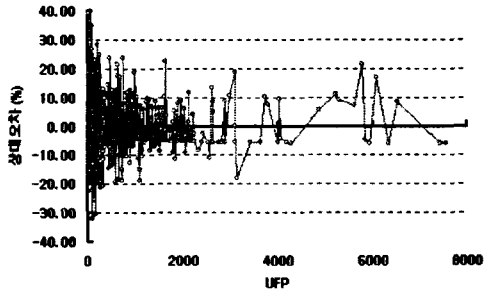
(a) Log(UFP) versus Log(AFP)



(b) Linear and Nonlinear Regression Model
[그림 2] 변수변환에 따른 회귀 모델
[Fig. 2] Regression Model According to Variable Transformation



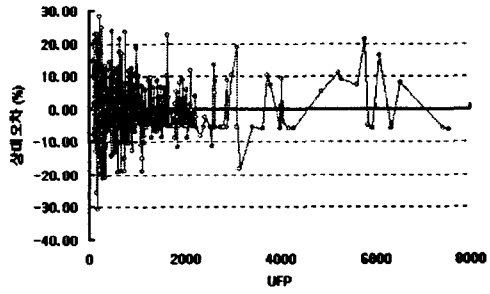
(a) $AFP = 0.9370UFP + 21.3255$ Model



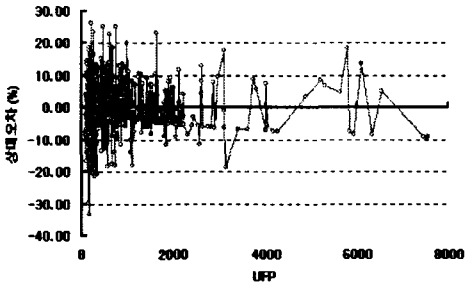
(b) $AFP = 1.2026UFP^{0.9688}$ Model

[그림 3] 상대오차

[Fig. 3] Relative Error



(a) $AFP = 0.9370UFP + 21.3255$ Model



(b) $AFP = 1.2026UFP^{0.9688}$ Model

[그림 4] $FP \leq 100$을 제외한 상대오차

[Fig. 4] Relative Error Except $FP \leq 100$

3.2 사례연구 II

다양한 요인들에 의해 시스템 수준의 복잡도가 변화될 수 있다. 그러나 이들 다양한 요인들 모두를 정확히 파악하는 것은 불가능하다. 또한, 다양한 요인들 상호간의 복잡한 관계를 모두 반영할 수 있는 모델을 구하는 것도 현실적으로 불가능하다. 따라서, 본 사례연구에서는 실제로 프로젝트를 수행할 때 프로젝트의 형태에 따라 시스템 수준의 복잡도를 반영한 기능점수를 계산하는 모델을 제시하고자 한다. 프로젝트 형태는 신규 개발 (New Development, ND), 성능을 향상시키기 위한 유지보수 (Enhancement, EH)와 재개발 (Re-development, RD)로 분류된다[10]. 그러나 다른 속성들을 고려할 수도 있다. 사례연구 I에서 가장 좋은 결과를 보인 비선형 로그변환 모델을 고려한다.

789개 프로젝트 중에 신규 개발 프로젝트는 421개, 유지보수 프로젝트는 317개이며, 재개발 프로젝트는 45개를 대상으로 하였다. 개발형태별 프로젝트에 대해 회귀분석을 수행한 결과는 [그림 5]에, 각 모델들은 식 (7)-(9)에 제시하였다.

$$AFP_{ND} = 1.2326UFP^{0.9655} \qquad (7)$$

$$AFP_{EH} = 1.1785 UFP^{0.9712} \quad (8)$$

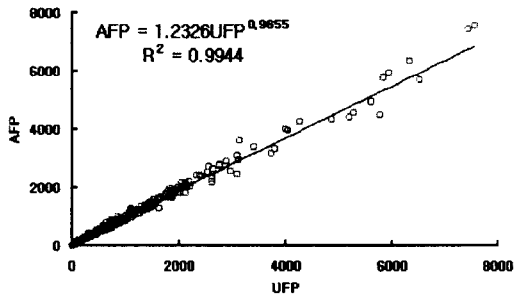
$$AFP_{RD} = 1.2596 UFP^{0.9633} \quad (9)$$

개발형태별 AFP 추정 모델의 성능은 <표 4>에, 상대오차는 [그림 6]에 제시되어 있다. 대략 기능점수가 150점 이하인 프로젝트에 대해서는 30% - 40%의 편차를 갖고 있으나 이보다 큰 규모의 소프트웨어인 경우에는 거의 대부분이 10% 이내의 편차를 가지고 있다.

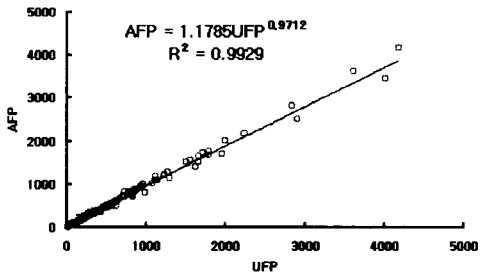
<표 4> 개발형태에 따른 모델 성능

<Table 4> Model Performance Accordance to Development Type

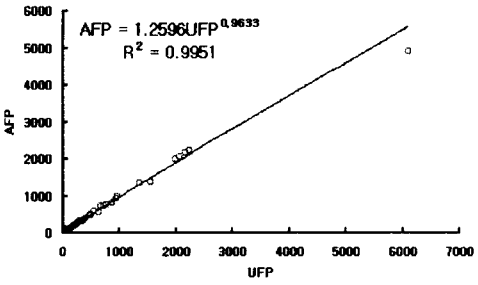
프로젝트형태	모델	성능	
		결정계수	MMRE
신규개발	$AFP_{ND} = 1.2326 UFP^{0.9655}$	0.9944	6.43%
유지보수	$AFP_{EH} = 1.1785 UFP^{0.9712}$	0.9929	5.64%
재개발	$AFP_{RD} = 1.2596 UFP^{0.9633}$	0.9951	4.66%



(a) New Developing Project



(b) Maintenance Project

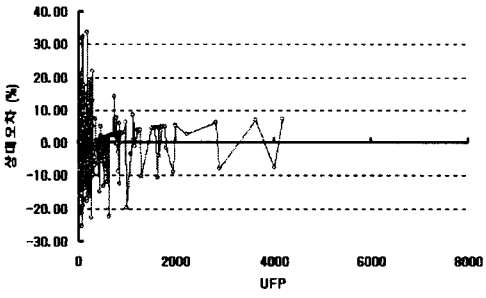


(c) Redeveloping Project

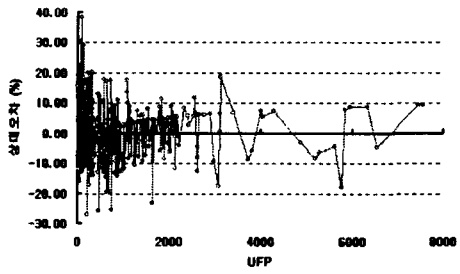
[그림 5] 개발형태에 따른 AFP 계산 모델

[Fig. 5] AFP Calculating Model Accordance to Development Type

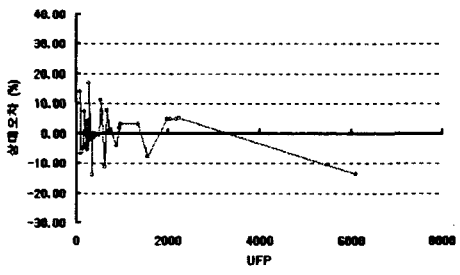
개발형태와 무관하게 또는 개발형태를 고려하여 적용할 경우 모두, 모델의 성능 측면에서는 결정계수나 평균 상대오차가 큰 차이를 보이지 않고 있다. 본 제안 모델들을 이용할 경우, UFP로부터 VAF를 고려하지 않고 직접 AFP를 측정시, 평균 6.5% 이하의 오차를 가지고 계산이 가능하다. 그러므로 과거 10년 이상의 개발 경험으로부터 획득된 대규모 프로젝트들로부터 경험적으로 유도된 본 모델을 적용시 VAF를 구하기 위해 소요되는 노력, 시간과 더불어 측정과정에서 발생할 수 있는 보다 큰 오차 발생 위험도 줄일 수 있는 장점이 있다.



(a) New Developing Project



(b) Maintenance Project



(c) Redeveloping Project

[그림 6] 개발형태에 따른 AFP 추정 모델의 상대오차

[Fig. 6]Relative Error of AFP Estimating Model Accordance to Development Type

4. 결론 및 향후 연구과제

기능점수를 계산하기 위해서는 소프트웨어 프로젝트의 기능을 구성하고 있는 5가지 기능요소 각각을 식별하고 복잡도를 계산하여 UFP를 계산한다. 여기에 시스템 수준의 복잡도인 VAF를 곱하여 최종적으로 기능점수 AFP가 계산된다. 그러나 이 과정에서 많은 시간과 노력이 소모될 수 있으며, 식별의 정확성도 저하될 수 있다. 또한, VAF를 계산시 시스템 전문가의 도움이 필요하다. 따라서, 본 논문은 VAF 계산 과정을 무시하고 UFP로부터 직접 AFP를 계산할 수 있는 모델을 회귀분석을 통해 제시하였다.

제시된 모델은 ISBSG Benchmark Release 6[10]의 최근 다양한 개발환경과 방법론으로 개발된 783개 프로젝트를 대상으로 하였다. 먼저 프로젝트 전체 데이터에 대한 선형과 비선형 모델을 비교하여 좋은 모델을 선정하였다. 다음으로, 개발형태별로 분류된 프로젝트별로 회귀모델을 제시하였다.

본 제안 모델을 적용시, UFP만 계산되면 시스템 전문가의 도움 없이 AFP를 계산할 수 있으며, 제안된 모델은 다음과 같은 장점을 가지고 있다.

(1) VAF를 계산하여 UFP와 곱한 값을 이용해 AFP를 얻는 기존 방 법에 비해 단순히 UFP로부터 AFP를 얻는 본 모델을 적용시 6.5% 이하의 상대오차를 가지고 AFP를 계산할 수 있다.

(2) 다양한 개발환경과 조건하에서 개발된 대용량의 표본으로부터 유도되어 일반적으로 적용 가능한 모델이다.

모델을 유도하기 위해 전체 데이터를 이용하는 경우와 동질성을 가진 서브데이타를 이용하는 경우를 고려하였다. 본 논문은 동질성을 갖는 서브데이타를 형성하기 위해 개발형태만을 고려하였다. 그러나, 기능점수에 영향을 미치는 속성으로는 ISBSG Benchmark Release 6[10]에서 제기한 바와 같이 개발형태 뿐만 아니라, 개발 플랫폼, 사용 언어 형태, DBMS 사용 여부, 개발기법 사용 여부, 프로그램 언어, 적용분야, 응용 형태 등 다양한 속성들이 복합적으로 영향을 미칠 수 있다. 따라서 보다 일반화된 모델을 유도하기 위해 추후 이 분야에 대한 연구를 수행할 예정이다.

참고문헌

[1] J. Verner and G. Tate, "A Software Size Model," IEEE Trans. on Software Eng., Vol. 18, No. 4, pp. 265-278, 1992.

[2] 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.

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

[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] "Function Point FAQ," Software Composition Technologies, Inc. 1997.

[6] M. Bradley, "Function Point Counting Practices Manual, Release 4.1," International Function Point Users Group(IFPUG), 1999.

[7] T.C. Jones, "Function-Point Counting Practics Manual, Release 4.1," ISBSG Ltd. 1999.

[8] 김우철 et al., "현대통계학," 영지출판사, 1994.

[9] S. D. Conte, H. E. Dunsmore and V. Y. Shen, "Software Engineering Metrics and Models," Menlo Park., CA: Benjamin Cummings, 1986.

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

박 석 규(Seok-Gyu Park)

1988년 4월 ~ 2001년 2월 진주산업대학교 전산실장
1990년 3월 ~ 1992년 8월 경남대학교 컴퓨터공학과(석사)
2001년 3월 ~ 현재 강원도립대학 컴퓨터응용과 조교수
관심분야 : 소프트웨어공학, 시스템 분석 및 설계, 멀티미디어

박 중 양(Joong-Yang Park)

1982년 연세대학교 상경대학 응용통계학과(학사)
1984년 한국과학기술원 산업공학과 응용통계전공(석사)
1994년 한국과학기술원 산업공학과 응용통계전공(박사)
1984년-1989년 경상대학교 전산통계학과 교수
1989년-현재 경상대학교 통계정보학과 교수, 컴퓨터정보통신연구소 연구원
관심분야 : 소프트웨어 신뢰성, 신경망, 선형통계모형, 실험 계획법 등