소프트웨어 개발 비용을 추정하기 위한 FFP 기반 모델

박 주 석† · 정 기 원††

요 약

소프트웨어 규모를 측정하기 위한 기존의 기능점수 기법은 관리정보 시스템에 적합하도록 구성되어 있으나 최근의 실시간 및 내장형 시스템의 적용성 확장에 따라 완전기능점수(FFP, Full Function Point) 기법이 제안되었다. 그러나, FFP 기반의 소프트웨어 규모 측정 방법에 관한 많은 연구는 이루어 지고 있으나, FFP로 측정된 소프트웨어 규모에 대한 개발비용을 추정할 수 있는 모델 연구는 미흡한 실정이다. 본 논문은 FFP로 실제 개발된 소프트웨어의 완전기능점수를 기반으로 소프트웨어 개발에 투입될 노력을 추정하는 선형 회귀분석 모델과 거듭제곱 회귀분석 모델을 평가하여 가장 적합한 모델로써 거듭제곱 모델을 선정하였다. 선정된 거듭제곱 모델을 적용할 경우 가장 근사치의 소프트웨어 개발 비용을 추정할 수 있음을 보였다.

A FFP-based Model to Estimate Software Development Cost

Ju Seok Park† · Ki Won Chong††

ABSTRACT

The existing Function Point method to estimate the software size has been utilized frequently with the management information system. Due to the expanding usage of the real-time and embedded system, the Full Function Point method is being proposed. However, despite many research is being carried out relating to the software size, the research on the model to estimate the development cost from the measured software size is inadequate. This paper analyzed the linear regression model and power regression model which estimate the development cost from the software FFP. The power model is selected, which shows its estimation is most adequate.

키워드 : 소프트웨어 규모(Software Size), 기능점수(Function Point), 완전기능점수(Full Function Point), 개발 비용(Development Effort)

1. 서 론

소프트웨어 프로젝트의 노력(Effort, E), 비용(Cost, C)과 일정(Schedule)을 적절히 추정하기 위해서는 소프트웨어 규모에 대한 정량화가 가장 중요한 요소로 일반적으로 인식되고 있다. 이 목적을 위해 일반적으로 적용되는 측도(Measure)가 코드 수(Line Of Code, LOC)와 기능점수(Function Point, FP)이다. LOC 측도는 길이(Length)로서 규모를 측정하는 방법으로 소프트웨어 개발이 종료되었을 때 정확한 라인 수를 측정할 수 있고, 사용되는 언어에 따라 달라진다는 근본적인 취약점(Limitation)을 갖고 있음이 소프트웨어 공학 분야에서 인식되어 왔다[1, 2]. Albrecht[3]는 사용자 관점에 기반을 두고 기능성(Functionality)으로 소프트웨어 규모를 정량화하는 방법을 제안하였다. Albrecht의 방법은 IFPUG (International Function Point User's Group) 방법으로 현재 널리 알려져 있고 산업계에서 적용되고 있다. 이 방법은 많은 조직에서 유용한 결과를 제공하고 있으나 1990년대 들어 MIS(Management Information System) 소프트웨어 이외의 분야에서는 적용하기가 어렵다는 취약점이 제기되기 시작하였다[4].

소프트웨어 측정방법의 새로운 세대를 설계하고 판로를 개척하기 위해 산업계의 지원을 받아 1998년에 COSMIC(Common Software Management International Consortium)이 설립되었다. 그 결과 MIS 영역인 데이터 관리 위주에서 제어관리 위주인 실시간 시스템과 내장형(Embedded) 소프트웨어에도 적용할 수 있도록 적용범위를 확장하였으며, 실용성이 입증되었다[5]. 이를 FFP(Full Function Point) 또는 COSMIC-FFP라 불리운다. 이 방법은 다층 또는 다단계로 연결된 소프트웨어 구조의 임의의 계층에 있는 규모도 측정할 수 있으나 복잡한 수학계산으로 구성된 알고리즘 위주의 소프트웨어에 대한 기능성을 측정하도록 설계되지는

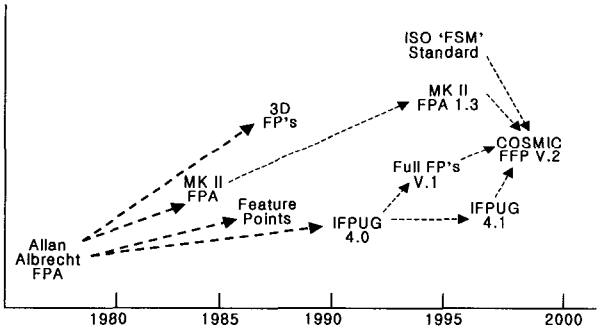
※ 본 연구는 숭실대학교 교내연구비 지원으로 이루어 졌음.
† 준 회 원 : 숭실대학교 대학원 컴퓨터학과
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
논문접수 : 2003년 7월 24일, 심사완료 : 2003년 9월 4일

못하였다. 2000년에 다양한 조직, 다양한 영역에서 개발되는 실시간과 내장형 소프트웨어에 대해 적용 타당성이 입증되었고, MIS 소프트웨어에 대해서도 유사한 결과를 얻었다. 또한 COSMIC-FFP는 소프트웨어 기능 규모 측정에 대한 ISO-14143 표준을 만족하도록 설계되었으며, 2003년 국제 표준화인 ISO-19761로 등재되었으며, ISBSG(International Software Benchmarking Standard Group)에 의해 인정되었다[4, 6].

FFP 기반의 소프트웨어 규모 측정 방법에 관한 많은 연구가 되고 있으나, FFP로 측정된 소프트웨어 규모에 대해 개발비용을 추정할 수 있는 모델 연구는 극소수에 불과한 실정이다. 따라서, 본 논문은 FFP로 측정된 소프트웨어 규모에 기반하여 개발에 투입될 비용을 추정할 수 있는 모델을 제시한다. 2장에서는 FP와 FFP에 대한 계산방법을, 3장에서는 FFP 기반의 개발비용 추정 관련 연구를 살펴본다. 4장에서는 FFP 기반의 개발비용 추정 모델을 제시하고 모델의 적합성을 평가해 본다.

2. FP vs. FFP

소프트웨어의 기능성에 기반하여 소프트웨어의 규모를 추정하기 위한 방법은 (그림 1)과 같이 발전하고 있다[7].



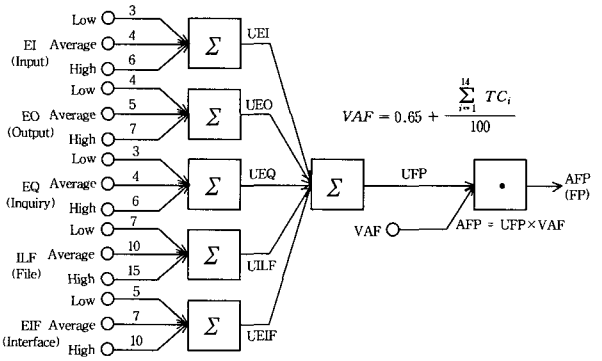
(그림 1) 기능 측면의 규모측정 방법 변천 과정

2.1 FP

조절이 안된 기능점수인 UFP(Unadjusted Function Pont)는 프로젝트 또는 응용프로그램이 사용자에게 제공하는 계산 가능한 기능성을 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[8]. 기능점수를 계산하는 복잡한 과정은 (그림 2)에 요약하여 제시하였다.

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 식별한다.



(그림 2) 기능점수 계산과정

[단계 2] 기능요소 형태 각각에 대해 <표 1>의 기준에 근거하여 복잡도 수준(Low, Average 또는 High)을 결정한다.

〈표 1〉 복잡도 수준

(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 1>에서 RET(Record Element Type)는 사용자가 인식할 수 있는 파일 또는 인터페이스에 있는 데이터 요소이

며, 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.

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

〈표 2〉 가중치 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

[단계 4] 사용자에게 제공되는 시스템 수준의 일반적인 처리 복잡도인 가치 조절 인자 VAF(Value Adjustment Factor)를 계산한다.

응용프로그램의 전반적인 기능성을 평가하는 일반적인 시스템 속성(General System Characteristics, GSC)들은 다음과 같이 14 종류가 있다. ① 데이터 통신. ② 분산 데이터 처리. ③ 성능. ④ 가중된 사용. 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 14 종류의 GSC 각각에 대한 영향 정도(Degree of Influence, DI)를 다음과 같이 점수화하여 평가 한다. ⓪ 관련 없거나 영향 없음. ① 우발적 영향. ② 보통의 영향. ③ 평균적 영향. ④ 중대한 영향. ⑤ 강력한 영향.

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

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

[단계 5] 소프트웨어 프로젝트에 대한 최종 조절된 기능점수인 AFP(Adjusted FP)는 UFP에 VAF를 곱하여 식 (4)로 계산된다.

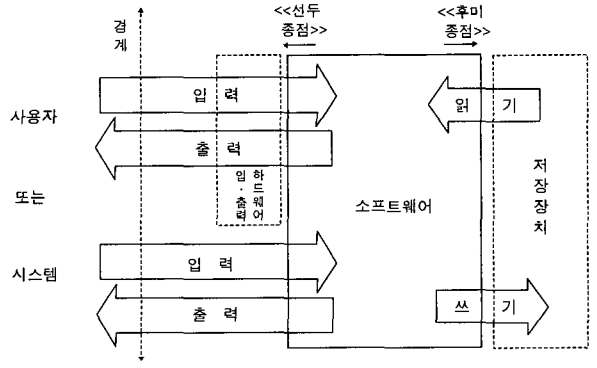
$$AFP = VAF \cdot UFP = VAF \cdot \sum_{i=1}^{5}\sum_{j=1}^{3} w_{ij} z_{ij} \tag{4}$$

식 (4)로 얻은 AFP를 일반적으로 기능점수라 부른다. 기능점수 기법의 기본원리는 비록 단순하고 간단하지만 이들 원리를 수 천 종류의 다양한 소프트웨어 프로젝트에 실제 적용하는 것은 결코 단순하지 않다. Jones[9]은 다른 개인들에 의해 계산된 기능점수의 차이가 150%(LOC의 경우와 동일) 이상 변동이 발생하면 유용한 측도(Metrics)로 고려할 가치가 없음을 제시하였다.

IFPUG(International Function Point User's Group) 지침서[8]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 인해 계산을 수행하는 시점을 계속 지연시키려고 하기 때문이다. 또 다른 이유는 시스템 전문가가 시스템의 특정한 특징만으로는 기능점수를 계산하는 것이 충분하지 않음을 제기하는 일이 종종 발생하기 때문이며, 기능점수를 계산하는 실무자는 이 특징을 VAF가 계산될 때 반영할 것을 요구하고 있다. 따라서, VAF는 전체적인 시스템의 복잡도를 파악해 주는 것으로 시스템 전문가의 도움을 받아 보다 정확한 계산을 할 수 있다[7]. 결과적으로 식 (3)에서 알 수 있듯이 VAF의 값에 따라 실제 계산된 AFP는 UFP에 대해 ±35%의 편차를 가진다. 따라서, VAF를 잘못 측정하였을 경우 최종적으로 계산된 소프트웨어 규모에 심한 오차를 유발시킬 수 있다.

2.2 FFP

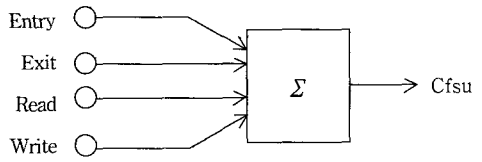
(그림 3)과 같이 FFP는 소프트웨어 시스템을 4종류의 데이터 이동 서브프로세스로 구별한다[4]. 소프트웨어 측면에서 바라볼 때, 입력(ENTRIES)와 출력(EXITS)는 사용자와 데이터 속성을 주고받는 것이며, 읽기(READS)와 쓰기(WRITES)는 저장장치와 데이터 속성을 주고받는 형태이다.



(그림 3) FFP 구성요소

FFP에 기반한 소프트웨어 규모의 단위는 Cfsu(Cosmic Functional Size Unit)로 표기한다. 소프트웨어 시스템의 임의의 i번째 계층에 대한 Cfsu는 식 (5)와 같이 계산되며, 계산과정은 (그림 4)에 제시되어 있다.

$$Size_{Cfsu}(layer_i) = \sum size(entries_i) + \sum size(exits_i) + \sum size(reads_i) + \sum size(writes_i) \quad (5)$$



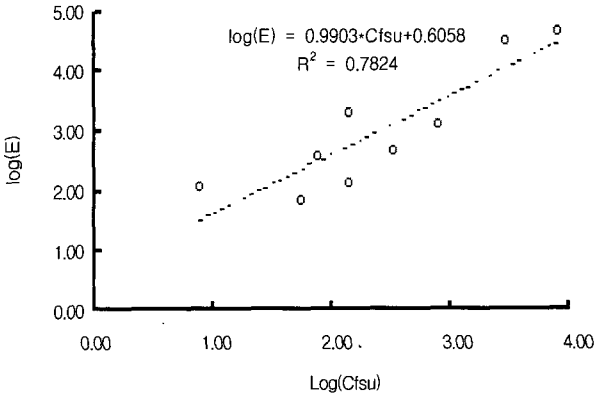
(그림 4) FFP 규모 계산 과정

3. FFP 기반 개발비용 추정 관련 연구

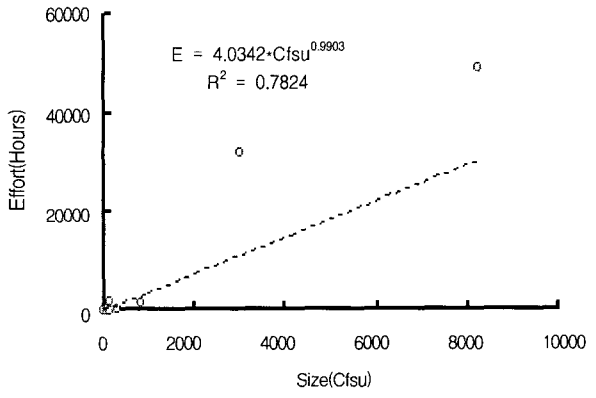
개발에 투입되는 총 노력(E)에 개발업체의 단위 시간당 소요되는 평균 비용을 곱하면 개발에 소요되는 직접 비용(C)이 산출되기 때문에 개발노력 추정 모델을 일명 개발비용 추정 모델이라고 칭한다. 그러므로 이후부터는 개발노력을 개발비용이라 칭한다. FFP 기반 개발비용 추정에 관한 연구로는 Abran et al.[10]과 Lévesque et al.[11] 모델이 있다. Abran et al[10]은 명세화(Specify), 구축(Build)과 시험(Test) 단계에 대한 개발비용를 추정하는 모델을 제시하였으며, Lévesque[11]는 총 개발비용에 대한 선형 모델을 제시하였다. 그러나 이 모델은 모델의 표현력인 결정계수가 61.61%에 불과하여 좋은 모델로 선정되기가 어렵다. 명세화 단계의 비용추정 모델을 E_S, 구축단계의 비용추정 모델을 E_B, 시험단계의 비용추정 모델을 E_T라 하자. Abran et al.[10]의 개발 각 단계에 소요되는 비용을 추정하기 위한 데이터는 <표 3>에 제시되어 있다. <표 3>의 데이터는 12개의 실시간 소프트웨어 프로젝트들로 3개 국가의 3개 개발회사로부터 1999년부터 2000년 기간동안 개발이 완료된 프로젝트들이다. 12개의 프로젝트들 중 2개는 유지보수 프로젝트이며, 나머지 10개는 신규로 개발된 프로젝트들임을 알 수 있다. <표 3>의 데이터들로부터 유도된 Abran et al.[10]의 명세화 단계 투입 비용 추정 모델 유도 과정은 (그림 5)에, 명세화, 구축과 시험 단계에 투입되는 비용을 추정하는 모델과 성능은 <표 4>에 제시되어 있다.

〈표 3〉 COSMIC-FFP 데이터 표본

프로젝트	형 태	개발언어	규 모 (Cfsu)	노력(Person-Hours)			
				명세화	구 축	시 험	계
A	개 발	SLEL	32	-	252	-	252
B	개 발	C++	76	381	1,457	401	2,239
C	개 발	C++	56	68	487	335	890
D	개 발	C	142	136	643	-	779
E	개 발	C	8	115	116	-	231
F	유지보수	C	142	2,060	1,487	5,055	8,602
G	개 발	C++	332	468	11,382	254	12,104
H	유지보수	Ada	624	-	15,815	1,372	17,187
I	개 발	Ada	810	1,304	10,903	4,548	16,755
J	개 발	Ada 95	484	-	20,808	6,772	27,580
K	개 발	Ada	8,251	49,000	66,000	93,000	208,000
L	개 발	Ada	3,004	32,000	27,000	20,000	79,000



(a) log 변수변환에 의한 개발비용 추정



(b) 변수 역변환을 거친 개발비용 추정 모델

(그림 5) 명세화단계 투입비용 추정 회귀분석 모델

〈표 4〉 개발단계별 투입비용 추정모델과 성능

단 계	모 델	결정계수	MMRE
명세화	$E_S = 4.0342\,Cfsu^{0.9903}$	0.7824	121.98
구 축	$E_B = 12.3110\,Cfsu^{1.015}$	0.8971	57.19
시 험	$E_T = 5.2124\,Cfsu^{1.0246}$	0.7019	119.31

모델을 평가하기 위해 결정계수 R^2와 MMRE(Mean Magnitude of Relative Error)가 적용되었다. 주어진 데이터들의 변동을 적절히 표현할 수 있는 모델을 찾기 위해 결정

계수를 이용한다. 종속변수의 값(E_S, E_B, $or\ E_T$)은 독립변수의 값($Cfsu$)에 의해 결정되는 부분과 미지의 오차의 합으로 나타나며, 총 변동(주어진 데이터의 실제 노력 값)중에서 회귀직선(E_S, E_B, $or\ E_T$)에 의해 설명되는 변동 비율을 결정계수(Coefficient of determination, $0 \le R^2 \le 1$)라 하며, 값이 클수록 쓸모 있는 회귀직선인 모델이 얻어진다[12]. 비록 결정계수에 의해 좋은 모델이 얻어졌더라도, 주어진 데이터들이 값이 큰 부분에 약간의 이상점들이 분포한 상태에서 회귀직선이 이 이상점들을 잘 표현하고 값이 작은 표본에 대해서는 표현을 하지 못할 경우에도 결정계수는 큰 값을 나타낼 수 있다. 이 경우 좋은 모델이라고 판단할 수 없다. 따라서, 모델 정확도(Accuracy)를 평가하는 측도로 MMRE가 적용되며, 다음과 같이 측정된다. RE = $\frac{\text{추정치} - \text{실측치}}{\text{실측치}}$, MRE(Magnitude of RE) = $|RE|$, MMRE (Mean MRE) = $100/n \times \sum_{i=1}^{n} MRE$. MMRE가 작은 값이면 모델은 평균적으로 좋은 모델임을 알 수 있다. Conte et al.[13]는 $MMRE \le 0.25(25\%)$이면 개발비용을 예측하는 모델로 채택 가능한 것으로 고려하였다.

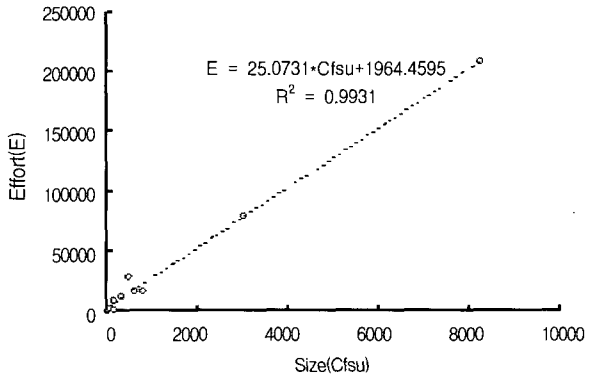
4. FFP 기반 개발비용 추정 모델

소프트웨어 개발에 소요되는 총 비용을 추정하는 모델을 E_A라 하자. Abran et al.[10]의 명세화, 구축과 시험단계에 투입되는 비용추정 모델은 개발 인력 또는 개발기간을 단계적으로 할당할 경우 적용될 수 있는 가치를 지니고 있다. 그러나 소프트웨어의 계획단계에서는 개발 각 단계별로 얼마만큼의 노력(비용)을 배정할 것인가에 관심을 갖기 보다는 개발에 소요되는 총 노력이나 비용이 보다 중요한 의사결정 요소이다. 또한, 개발 각 단계별로 투입되는 비용추정 모델을 적용하여 추정한 값들을 합하여 개발 단계에 투입되는 총 비용을 추정하는 경우($E_S+E_B+E_T$)는 개발에 소요되는 총 비용을 추정하는 모델 E_A을 적용할 때보다 큰 편차를 나타낼 수 있다. 이는 개발 각 단계에 투입되는 비용을 추정하는 모델들 각각이 갖고 있는 편차들이 합산되어 보다 큰 편차를 유발시킬 수 있기 때문이다. 결과적으로 소프트웨어 관리자 측면에서는 개발 각 단계별로 투입되는 비용 추정 모델보다는 개발에 소요되는 총 비용을 추정하는 모델이 보다 절실히 요구될 수 있으므로 본 장에서는 개발에 투입되는 총 비용을 추정하는 모델을 제시한다.

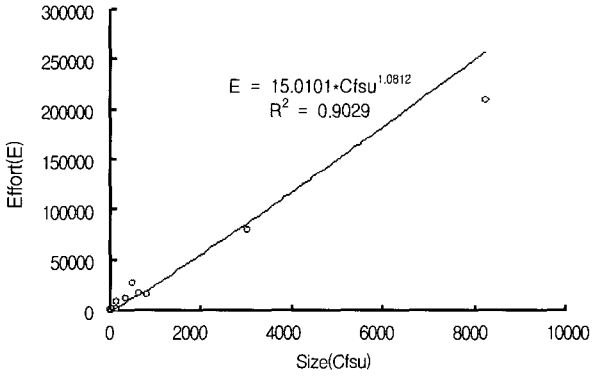
<표 3>의 12개 프로젝트들은 명세화, 구축과 시험 단계를 모두 거친 프로젝트들과 특정 단계만을 수행한 프로젝트들로 혼합되어 있어 12개 프로젝트 모두를 적용하여 소프트웨어 규모인 Cfsu에 대한 개발에 소요되는 총 노력을 추정하는 모델은 일반화된 결과를 유도하지 못할 수 도 있어 본 장에서는 참고자료로 활용할 수 있도록 제시하고, 이어서 명세화, 구축과 시험 단계 모두를 수행한 프로젝트들에 적용되는 개발비용 추정 모델을 제시한다.

4.1 특정 단계를 수행하지 않은 프로젝트도 고려한 경우

<표 3>의 12개 프로젝트를 대상으로 소프트웨어 규모인 Cfsu에 대한 개발 총 비용 관계에 대해 회귀분석을 수행한 결과는 (그림 6)에 제시하였다. (그림 6)의 FFP에 따른 개발비용 추정 모델에 대한 성능 분석 결과는 <표 5>에 제시되어 있다. 또한, 두 모델의 상대오차를 비교 분석한 결과는 (그림 7)에 제시하였다.



(a) 선형 회귀 모델



(b) 거듭제곱 회귀 모델

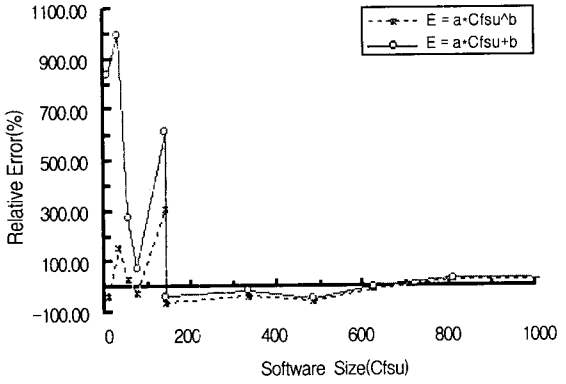
(그림 6) 총 비용추정 모델 회귀분석(전체데이터 이용시)

〈표 5〉 FFP 기반 총 개발비용 추정 모델(E_A) 성능

구 분	개발에 소요되는 총 비용추정 모델	결정계수	MMRE
선 형	$E_A = 25.0731 \cdot Cfsu + 1964.4595$	0.9931	244.45
거듭제곱	$E_A = 15.0101 \cdot Cfsu^{1.0812}$	0.9029	64.87

FFP 기반 비용추정 모델 분석 결과 결정계수 측면에서는 선형모델이 거듭제곱 모델에 비해 좋은 반면 MMRE 측면에서 좋지 못한 것으로 판명되었다. 즉, 선형회귀 모델은

소프트웨어의 규모가 작은 부분의 대부분의 데이터들에 부적합함에도 불구하고 규모가 큰 일부 데이터에 일치됨에 따라 모든 데이터에 대한 평균 절대오차가 작아 결정계수는 좋은 결과를 나타낸 반면 모든 데이터에 대한 평균 상대오차 측면에서는 규모가 작은 소프트웨어들에 대한 편차들이 반영되어 MMRE가 나쁜 결과를 얻었다. 이에 반해 거듭제곱 회귀 모델은 선형회귀 모델과 상반되는 결과를 나태내고 있다. 즉, 대부분의 데이터들이 규모가 작은 소프트웨어 부분에 보다 적합하고 최대 규모의 소프트웨어에서만이 큰 편차를 나타내는 결정계수는 나쁜 반면, MMRE는 좋은 결과를 나타내고 있다. 따라서, FFP 기반 총 비용추정 모델(E_A)은 거듭제곱모델을 적용하는 것이 보다 타당해 보인다.



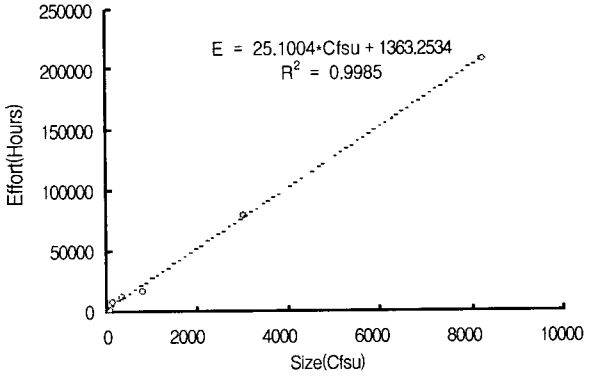
(그림 7) 상대오차 비교

4.2 명세화, 구축과 시험단계를 모두 수행한 경우

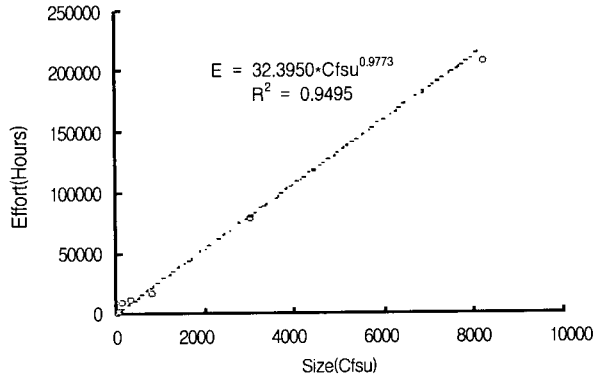
보다 타당한 FFP 기반 개발단계 소요 총 비용을 추정하는 모델을 제시하기 우해 <표 3>의 12개 프로젝트 중 명세화, 또는 시험단계를 수행하지 않은 A, D, E, H, J 프로젝트를 제외한 데이터들을 <표 6>에 다시 표기하였다. <표 6>의 데이터를 이용하여 개발에 소요되는 총 비용을 추정하는 모델을 얻기 위해 회귀분석을 수행한 결과는 (그림 8)에 제시되어 있다. 본 모델을 E_{SBT}라 하자. (그림 8)의 결과로 얻은 모델의 성능은 <표 7>에 제시되어 있다. E_{SBT} 모델도 E_A과 동일하게 결정계수 측면에서는 선형모델이 거듭제곱 모델에 비해 좋은 반면 MMRE 측면에서 좋지 못한 것으로 판명되었다. 그러나 특정 개발단계를 수행하지 않은 프로젝트까지 고려한 <표 5>와 비교시 선형이나 거듭제곱 모델 모두 월등한 성능 향상을 보임을 알 수 있다.

〈표 6〉 명세화, 구축과 시험단계를 모두 수행한 프로젝트

프로젝트	형 태	규 모 (Cfsu)	노력(Person-Hours)			
			명세화	구 축	시 험	계
B	개 발	76	381	1,457	401	2,239
C	개 발	56	68	487	335	890
F	유지보수	142	2,060	1,487	5,055	8,602
G	개 발	332	468	11,382	254	12,104
I	개 발	810	1,304	10,903	4,548	16,755
K	개 발	8,251	49,000	66,000	93,000	208,000
L	개 발	3,004	32,000	27,000	20,000	79,000



(a) 선형 회귀 모델



(b) 거듭제곱 회귀 모델

(그림 8) 총 비용추정 모델 회귀분석(모든 개발단계 수행 데이터만 이용시)

〈표 7〉 FFP 기반 총 개발비용 추정 모델(E_{SBT}) 성능

구 분	개발에 소요되는 총 비용추정 모델	결정계수	MMRE
선 형	$E_{SBT} = 25.1004 \cdot Cfsu + 1363.2534$	0.9985	50.33
거듭제곱	$E_{SBT} = 32.3950 \cdot Cfsu^{0.9773}$	0.9495	28.95

모델의 성능을 비교평가 한 결과, 결론적으로 FFP 기반 소프트웨어 개발에 소요되는 총 비용을 추정하기 위한 모델로 $E_{SBT} = 32.3950 \cdot Cfsu^{0.9773}$를 제안한다.

추가적으로, 소프트웨어를 개발할 경우, 개발 각 단계에 얼마만큼의 노력을 할당할 것인가에 따라 프로젝트의 성공을 좌우할 수 있다. 이와 관련하여 <표 6>의 데이터를 이용하여 분석한 결과는 <표 8>에 제시하였다. 결론적으로 명세화 단계 : 구축 단계 : 시험단계에 대한 노력 할당 비율은 대략 20 : 50 : 30임을 알 수 있다. 따라서, 개발 일정과

개발 인원을 배분하는 데 있어 경험법칙(Rules of Thumb)으로 이 지침을 적용할 수 있을 것이다.

〈표 8〉 개발 단계별 노력할당 비율

프로젝트	형 태	투입 노력(Person-Hours)			비 율(%)		
		명세화 단 계	구 축 단 계	시 험 단 계	명세화 단 계	구 축 단 계	시 험 단계
B	개 발	381	1,457	401	17.02	65.07	17.91
C	개 발	68	487	335	7.64	54.72	37.64
F	유지보수	2,060	1,487	5,055	23.95	17.29	58.77
G	개 발	468	11,382	254	3.87	94.04	2.10
I	개 발	1,304	10,903	4,548	7.78	65.07	27.14
K	개 발	49,000	66,000	93,000	23.56	31.73	44.71
L	개 발	32,000	27,000	20,000	40.51	34.18	25.32
평 균					**17.76**	**51.73**	**30.51**

5. 결론 및 향후 연구과제

소프트웨어 공학 분야에서 소프트웨어를 개발하는데 소요되는 노력, 비용과 일정을 추정하기 위해 필수적으로 요구되는 속성인 소프트웨어 규모를 어떻게 정량화 할 것인가에 대한 많은 연구가 진행되고 있다. 초창기에는 길이로서 소프트웨어의 규모를 정량화 하는 방법으로 라인 수를 이용하였으나, 프로젝트의 성공적인 완수를 위해 프로젝트 계획단계에서 개발에 소요되는 노력, 비용과 일정을 추정하는 것이 필요하게 되었다. 그러나 동일한 프로그램이라 하더라도 적용되는 언어에 따라 라인 수가 다르게 나타나며, 요구사항 분석 단계에서 정확한 추정이 어려운 문제점을 갖고 있어, 사용자에게 납품되는 기능성 또는 소프트웨어의 복잡성을 고려한 규모측정 방법이 연구되고, 산업계에 적용되고 있다. 소프트웨어의 기능성에 기반한 규모측정 방법으로 기능점수가 있으며, 이는 초창기에는 관리정보시스템에 적합하도록 개발된 방법으로 점차 실시간과 내장형 소프트웨어에도 적용할 수 있도록 발전하였으며, 이를 FFP라 칭한다.

본 논문은 FFP에 기반하여 측정된 소프트웨어 규모인 Cfsu를 이용하여 소프트웨어 개발에 소요되는 총 비용을 추정하는 모델을 제시하였다. 다양한 방법으로 분석을 하고, 회귀분석을 통해 적절한 모델을 유도하였으며, 모델의 적합성도 평가하였다. 결론적으로 개발의 각 단계를 모두 수행한 프로젝트에 적합한 거듭제곱모델이 가장 적합한 비용추정 모델로 선정되었으며, Conte et al.[13]의 비용 예측 모델로서의 채택 기준에 근접하는 결과도 얻을 수 있었다.

FFP 기반 소프트웨어 비용추정 모델을 유도하는데 적용된 데이터는 다양한 개발업체의 다양한 환경에서 적용된 대용량의 데이터를 이용해 일반화된 모델을 유도하는 것이 필요하다. 그러나, FFP는 최근 들어 국제 표준화가 되었으며, 현재는 개발현장에서 적합성이 검증되고 있는 단계로 실제적으로 많은 데이터가 존재하지 않는다. 일 예로, FP나 FFP 기법을 보완 발전시키고 있는 ISBSG[14]에서도 2003년 현재 2,027건의 프로젝트 데이터 중 불과 20건 만이 FFP 관련 자료이고 나머지는 FP와 관련된 자료이다. 따라서, 추후 대용량의 데이터를 수집하여 일반화된 FFP 기반의 소프트웨어 비용추정모델 제시와 더불어, 개발일정, 개발 팀의 규모도 추정할 수 있는 모델에 대한 연구가 수행될 것이다.

참 고 문 헌

[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 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.

[4] ISO TC JTC1, "Software Engineering - COSMIC-FFP Functional Size Measurement Method," ISO, 2001.

[5] F. Bootsma, "Applying Full Function Points to Drive Strategic Business Improvement with the Real-Time Software Environment," Annual IFPUG Conference, New Orleans, 1999.

[6] HierarchyMaster, "Software Metrics," http://www.hmaster.com/FFP/metrics.html, HierarchyMaster, Pty, Ltd., Australia, 2003.

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

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

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

[10] A. Abran, C. Symons, and S. Oligny, "An Overview of COSMIC-FFP Field Trial Results," ESCOM 2001, London, England, 2001.

[11] G. Levesque and V. Bevo, "Comparing COSMIC-FFP and SLIM Back-Firing Function Points Size Measurements," ICSSEA, 2001.

[12] 김우철 et al., "현대통계학", 영지출판사, 1994.

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

[14] ISBSG, "Worldwide Software Development - The Benchmark Release 8," Victoria, Australia International Software Benchmarking Standards Group, 2003.

박 주 석

e-mail : iage2k@dreamwiz.com

1984년 해군사관학교 전자공학과(공학사)

1995년 국방대학원 전자계산학과(전산학석사)

2001년~현재 숭실대학교 일반대학원 컴퓨터학과 박사과정 수료

관심분야 : 비용산정, 표준 및 프로세스, 정보체계사업관리, 소프트웨어품질보증, 정보전

정 기 원

e-mail : chong@comp.ssu.ac.kr

1967년 서울대학교 전기공학과(공학사)

1981년 미국 알라바마주립대 전산학과(전산학석사)

1983년 미국 텍사스주립대 전산학과(전산학박사)

1966년~1968년 미8군(IBM 기계정비 담당)

1971년~1975년 한국과학기술연구소

1975년~1990년 국방과학연구소(책임연구원)

1990년~현재 숭실대학교 컴퓨터학부 교수

관심분야 : 소프트웨어 프로세스, 소프트웨어 개발방법론, 정보시스템, 전자거래(CALS/EC)