소프트웨어 규모 산정을 위한 개선된 기능 점수 측정 모델☆

Improved Function Point Measurement Model for Software Size Estimation

정 인 용* 우 덕 제** 박 진 형*** 정 창 성****
Jung, In Yong Woo, Doug Je Park, Jin Hyeong Jeong, Chang Sung

요 약

소프트웨어 규모 추정은 소프트웨어 Life-Cycle 초기에 분석되어 규모와 비용의 예측에 도움을 주어야 한다. 2004년 소프트웨어 사업대가 기준에 국제표준에 기반한 기능점수 방식이 도입된 후 사용자 입장에서 소프트웨어의 규모를 바라보고 비용을 산정하는 기반이 마련되었다. 그러나 현재의 기능 점수 측정 방식은 익숙하지 않은 일반 사용자가 접근하기 쉽지 않고, 모든 시스템 및 기능의 복잡도 가중치가 획일화 되어 있어 내부 계산 로직이 복잡한 공학용 소프트웨어나 과학계산용, 시뮬레이션 소프트웨어에 대한 산정 방식에서 그 규모를 적절히 산정하지 못하는 문제점을 안고 있다. 본 논문에서는 기존의 기능점수 측정 절차를 간략화하고 프로젝트 초기에 규모의 추정을 쉽고 빠르게 수행할 수 있는 모델을 제시한다. 또한 특정 조직의 특성을 반영할 수 있는 수학적 가중치 산출 모형을 제시함으로써 고정된 복잡도 가중치에 대한 논란의 여지를 없애고 조직의 데이터가 쌓일수록 해당 조직의 특성을 반영해 나갈 수 있는 수학적 가중치 산출 모형을 제시한다. 제시한 모델은 평가 결과 기존의 FPA(Function Point Analysis) 방식보다 빠르게 규모를 측정할 수 있고 LOC(Line of Code)와의 상관관계도 더 높은 장점이 있다.

Abstract

A software size estimation has to be analyzed in the beginning of the software life-cycle and helpful to the prediction of its size and cost. The software cost has been calculated by estimating software size from the user's point of view since the function point method based on international standards was introduced for the estimation of software size in 2004. However, the current function point method is not easy to be exploited for unfamiliar user, and has a problem that it cannot estimate the proper size for software such as engineering software, scientific calculations and simulation with complicated internal computational logic. This paper presents an improved model which can simplify the existing function point measurement procedure, and perform the estimation of software size in easy and fast way at the initial stage of project. Moreover, it presents a mathematical weighted value calculation model which can solve the problem of the fixed complexity weighted value and reflect the characteristics of organization as its data is pilled up. Our evaluation shows that the presented model has advantage that it can measure the size more rapidly than the existing FPA methods and has more correlation with LOC.

☞ keyword : Software Size Estimation, FPA, 기능점수 측정 모델

1. 서 론

소프트웨어 규모 예측은 프로젝트 초반에 수행되어 해당 프로젝트의 비용, 기간, 자원 등의 결정에 영향을 미치며 정확한 규모 예측은 고품질의 소프트웨어 구현과 성공적인 프로젝트 완료를 위한 선행 과제이다. 소프트웨어 규모를 추정하는 기법으로 가장 널리 사용되는 방법은 프로그램

* 준 회 원 : 고려대학교 대학원 전자전기공학과(공학석사)
dekarno@korea.ac.kr
** 정 회 원 : 한전KDN
woosaint@korea.ac.kr
*** 준 회 원 : 고려대학교 대학원 전자전기공학과(공학석사)
kanonerin@korea.ac.kr
**** 정 회 원 : 고려대학교 전자공학과 정교수
csjeong@korea.ac.kr
[2009/01/07 투고 - 2009/01/23 심사 - 2009/02/16 심사완료]
☆ 이 논문은 서울시 연구 비즈니스 개발 프로그램, 정보통신연구진흥원 ITRC 지원 프로그램, 고려대학교 BK21사업단 지원에 의한 연구이었음(This work was supported by the Seoul Research and Business Development Program of Seoul, Korea, the ITRC support program of IITA, a Brain Korea 21 project and a grant of Korea University)

라인 수(Line Of Code, 이하 LOC) 방식과 FSM(Functional Size Measurement) 방식이 있다.

LOC 방식은 해당 소프트웨어의 프로그램 라인 수를 측정하는 방식으로 개발 규모를 산정하는 데 그 의미가 명확하여 널리 쓰이고 있다. 하지만 개발 초기 단계에서의 프로그램 라인 수를 산정할 수 없는 것과 객체지향 방법론, CBD 방법론 등 진보된 개발 방법론과 개발자의 능력 등 환경에 따라 결과가 달라지는 단점을 가지고 있다.

FSM 방식은 사용자 관점에서 소프트웨어가 제공하는 기능을 분석하고 이를 정량화하여 규모를 측정하는 방식이다. 1970년대 말 IBM의 Allan J. Albrecht에 의해 최초로 고안된 기능점수 분석(Function Point Analysis, 이하 FPA)방식은 FSM 방식의 모태이며 개발이 완료된 후 측정 가능한 물리적인 크기 대신 초기 단계에서 고객이 요구하는 시스템의 기능을 크기로 해당 소프트웨어의 규모를 측정하는 방식이다[1]. 현재 FSM 방식은 국제 표준(ISO/IEC 14143) 방식이며 해당 표준을 따르는 방법으로는 FPA, NESMA, MarkII, COSMIC 등이 있다. 전 세계 많은 나라에서 FSM 방식을 통해 소프트웨어의 규모를 산정하고 있으며 우리나라에서는 2004년 소프트웨어 사업대가 기준에서 소프트웨어 규모 산정 방식을 기존의 본수 방식에서 기능점수 방식으로 전환하였다[2].

FPA 방식 도입 이후 주로 공공사업과 대규모 프로젝트를 시작으로 기능점수 방식을 도입하고 있지만 FPA 측정 방식이 가지고 있는 사용자 주관적이라는 모호성과 다양한 소프트웨어 기능의 복잡도를 고정된 복잡도 매트릭스로 표현하는 한계, 기능점수 측정 경험 부족 등으로 활발히 쓰이지 못하는 실정이다.

본 논문에서는 FPA 방식이 가지고 있는 문제점을 분석하고 이를 개선할 수 있는 모델을 제시하여 기존의 FPA 방식보다 효율적이고 효과적인 측정 방식을 구현한다.

2. 관련 연구

2.1 FPA (Function Point Analysis)

FPA 기법은 FSM 기법 가운데 전 세계적으로 가장 널리 쓰이는 기법 중 하나로 1979년 IBM의 Allan Albrecht에 의해 고안되었으며 1984년 'Albrecht 1984'로 불리우는 "IBM CIS & A Guideline 313, AD/M Productivity Measurement and Estimate Validation"를 통해 대중에게 소개되었다[3]. 이후 IFPUG(International Function Point Users Group)에 의해 개선되며 2002년 조정인자를 제외한 4.1버전이 ISO/IEC 표준 규격을 획득하였으며 2004년 CPM(Counting Practice Manual) 4.2 버전이 발간되었다[4]. 기능점수 분석은 사용자 관점에서의 소프트웨어 개발 규모를 측정하기 위한 표준 기법으로 논리적 설계에 기초하여 사용자에게 제공되는 소프트웨어 기능들을 정량적으로 측정하는 것으로서 다음과 같은 목적을 가지고 있다[5].

① 사용자가 요구하여 제공 받는 기능들을 측정
② 구현 기술과는 무관하게 소프트웨어 개발 및 유지보수 규모를 측정

기능점수 분석 기법은 사용자 관점에서 규모를 산정하기 때문에 기존의 LOC나 경험치에 의한 규모 추정 방식에서는 기대할 수 없었던 많은 장점을 가지고 있다. 예를 들어 소프트웨어 패키지 도입 시 해당 제품의 물리적 크기나 내부적 구현을 고려하지 않고 오직 사용자에게 유용한 기능만을 식별하기 때문에 사용자 입장에서는 일관된 도입 기준을 가지는 셈이다. IFPUG의 CPM 4.2는 기능점수 분석의 용도를 다음과 같이 기술하고 있다[5].

① 패키지의 모든 기능을 측정함으로써, 구매한 애플리케이션 패키지의 규모를 결정하는 도구
② 사용자의 요구를 만족시키는 특정 기능을

측정함으로써, 애플리케이션 패키지가 사용자 조직에 유익한지의 결정을 돕는 도구

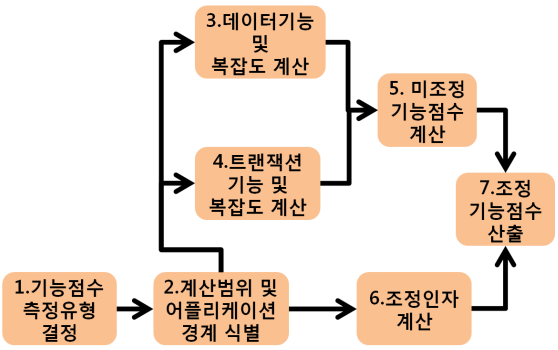
③ 품질과 생산성 분석을 돕기 위해 소프트웨어 제품 단위를 측정하는 도구

④ 소프트웨어 개발 및 유지보수에 필요한 비용과 자원을 측정하기 위한 도구

⑤ 소프트웨어 비교를 위한 정규화 요소

위와 같은 이점과 더불어 전 세계적으로 표준이 되는 측정 도구를 사용함으로써 유용한 통계 자료와 측정 데이터 비교 등 소프트웨어 생산성 및 품질 향상에 많은 도움을 받을 수 있다.

기능점수 측정 절차는 그림 1과 같이 7 단계를 거쳐 이루어진다.



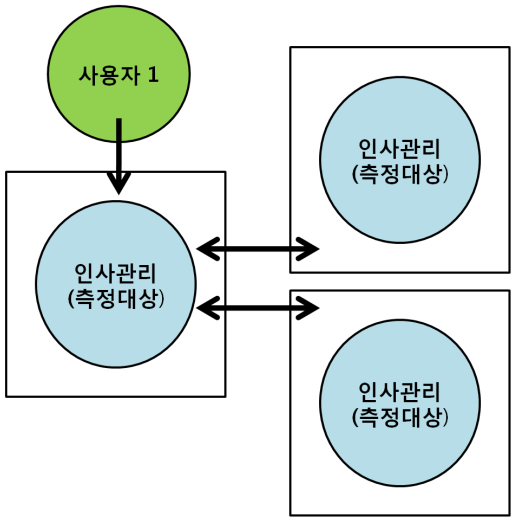
(그림 1) FP 측정 절차

① 단계 1 : 측정 유형 결정

- 개발 프로젝트(DFP : Development Function Point) : 프로젝트가 종료되어서 인도된 소프트웨어의 최초 설치와 함께 사용자에게 제공된 기능을 측정
- 개선 프로젝트(EFP : Enhancement Function Point) : 기존 애플리케이션의 변경 부분(추가, 수정, 삭제)을 측정
- 애플리케이션 (Application) : 베이스라인 또는 설치된 기능점수 측정이라고도 부르며 개발 프로젝트 기능점수 측정이 완료되었을 때 초기화되고, 개선 프로젝트의 종료로 애플리케이션 기능이 변경될 때마다 수정

② 단계 2 : 측정범위와 애플리케이션 경계설정

- 규모 측정 대상 소프트웨어의 집합을 정의
- 측정 대상 소프트웨어와 사용자 간의 경계를 식별



(그림 2) 애플리케이션 경계

③ 단계 3 : 데이터 기능 측정

- 사용자의 내부 및 외부 데이터 요구사항을 충족시키기 위해 제공되는 기능 측정
- ILF (Internal Logical File), EIF (External Interface File) 식별
  - ILF : 사용자가 식별할 수 있는 논리적으로 연관된 데이터 그룹 또는 제어정보로 애플리케이션 내부에서 유지(추가・수정・삭제)
  - EIF : 사용자가 식별할 수 있는 논리적으로 연관된 데이터 그룹 또는 제어정보로 다른 애플리케이션의 경계 내부에서 유지(추가・수정・삭제)

④ 단계 4 : 트랜잭션 기능 측정

- 애플리케이션 데이터를 처리하여 사용자에게 제공하는 기능

- EI(External Input), EO(External Output), EQ(External InQuery) 식별
  - EI : 애플리케이션 경계 밖에서 들어오는 데이터나 제어 정보를 처리하는 단위 프로세스
  - EO : 데이터나 제어 정보를 애플리케이션 경계 밖으로 보내는 단위 프로세스로서 처리로직을 통해 사용자에게 정보를 제공
  - EQ : 데이터나 제어 정보를 애플리케이션 경계 밖으로 보내는 단위 프로세스로서 논리파일로부터 사용자에게 정보를 제공

⑤ 단계 5 : 미조정 기능점수 계산
- 데이터기능 점수와 트랜잭션 기능 점수의 합

⑥ 단계 6 : 조정인자(Value Adjustment Factor, 이하 VAF) 결정
- 미조정 기능 점수를 ± 35% 범위에서 조정
- VAF = (TDI * 0.01) + 0.65

⑦ 단계 7 : 조정 기능점수 계산
- 미조정 기능점수에 조정인자를 곱하여 산정

데이터 기능 측정과 트랜잭션 기능 측정 시 각 기능 유형 별 복잡도를 부여하게 된다. 복잡도는 데이터 기능의 경우 레코드 요소 유형(RET, Record Element Type)의 개수와 데이터 요소 유형(DET, Data Element Type)의 개수에 따라, 트랜잭션 기능의 경우 참조하는 데이터 기능인 FTR(File Type Referenced)과 DET의 개수에 따라 '낮음', '보통', '높음'의 세 가지로 분류된다. 각 기능 유형 별 복잡도와 복잡도에 따른 가중치는 아래 표와 같다.

(표 1) ILF, EIF 타입 기능점수 내부 복잡도

레코드요소	데이터요소유형의 개수		
유형의 개수	1~19	20~50	51이상
1	낮음	낮음	보통
2~5	낮음	보통	높음
6이상	보통	높음	높음

(표 2) EI 타입 기능점수 내부 복잡도

참조파일	데이터요소유형의 개수		
유형의 개수	1~4	5~15	16이상
0~1	낮음	낮음	보통
2	낮음	보통	높음
3이상	보통	높음	높음

(표 3) EO, EQ 타입 기능점수 내부 복잡도

참조파일	데이터요소유형의 개수		
유형의 개수	1~5	6~19	20이상
0~1	낮음	낮음	보통
2~3	낮음	보통	높음
4이상	보통	높음	높음

(표 4) 각 기능점수의 내부 복잡도 별 가중치

참조파일 유형의 개수	복잡도		
	낮음	보통	높음
ILF	x 7	x 10	x 15
EIF	x 5	x 7	x 10
EI/EQ	x 3	x 4	x 6
EO	x 4	x 5	x 7

2.2 NESMA

NESMA (Netherlands Software Metrics Association)은 Detailed, Estimated, Indicative의 3가지 측정 방식을 정의하였다[6].

① Detailed Count
② Estimated Count
③ Indicative Count (the "Dutch Method")

Detailed Count 방식은 일반시스템특성(GSC, General System Characteristics)이 적용되지 않는 것을 제외하고는 IFPUG의 기능점수 산정방식과 동일하다. Estimated Count 방식은 각 기능 유형에 default 가중치를 부여한다. Indicative Count 방식은 데이터 기능 유형(ILF, EIF)의 개수만을 고려하므로 그다지 정확하지 않은 추정을 얻기 위해 필요한 공수의 최소화에 초점을 맞춘 방식이다.

2.3 Mark II

Charles Symons는 기능 측정 방법론 중의 하나인 British Mark 2 기법을 개발함으로써 명성을 얻었다. 그는 1998년에 자신의 방법론을 발표했으며, 여기서 기능 점수 방법론의 단점을 언급하고 기존 버전의 기능 점수에 대해 다음과 같은 수정을 제안했다[7].

① 데이터 구조를 통과해서 움직일 때의 엔티티의 수와 그 성능을 측정함으로써, 파일을 다루는 데 있어서의 주관성을 줄인다.
② 애플리케이션 경계와 상관없이 단일 시스템으로서 또는 관련된 서브시스템의 집합으로 동일한 합계를 계산하도록 기능 점수 방법을 수정한다.
③ 사용자에게 전달되는 가치보다는 그 기능을 생산하는 데 드는 노력에 초점을 맞춘다.
④ 14개의 일반 시스템 특성에 6개의 복잡도 요소를 추가한다.

Mark 2 방법은 영국에서 주로 활용되고 있다.

3. 기존 FPA 방식

본 장에서는 그동안 논란이 되어왔던 기존 FPA 방식의 문제점들을 분석한다.

3.1 논리파일 측정

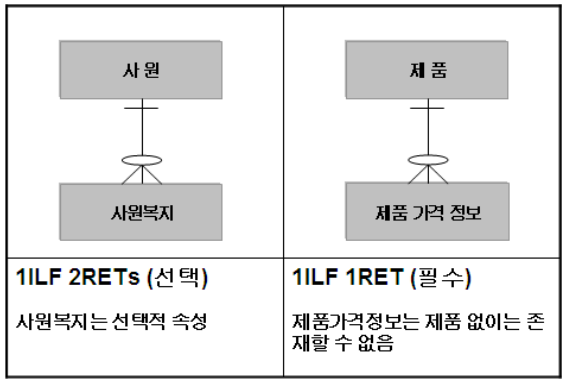
논리파일 (Logical File)의 측정 절차는 다음과 같다.

① Step 1 : ILF와 EIF의 식별
② Step 2 : 미조정 기능점수에 ILF 또는 EIF의 복잡도 및 가중치 산정

논리파일의 기능점수를 측정 과정에서 가장 많은 어려움을 겪는 부분이 논리파일의 복잡도를 산정하는 과정이다. 식별된 모든 ILF와 EIF에 대해 각각의 DET와 RET의 수에 따라 복잡도를 할당하도록 되어있는데[5], 사용자의 관점에서 논리적인 데이터 그룹으로 RET를 식별하는 과정에서 많은 해석의 차이를 보이고 있다. RET는 ILF나 EIF 안에서 사용자 식별 가능한 데이터 요소의 서브그룹으로 다음과 같은 두 가지 유형이 있다.

① 선택적 (Optional)
② 필수적 (Madatory)

선택적 서브그룹이란 사용자가 데이터의 인스턴스를 추가 또는 생성하는 단위 프로세스에서 서브 그룹을 사용할 수도 있고 사용하지 않을 수도 있는 데이터 그룹을 의미하며 필수적 서브그룹은 사용자가 적어도 하나 이상을 사용해야하는 서브그룹을 의미한다.



(그림 3) RET 산정 예제

3.2 VAF (Value Adjustment Factor)

조정인자(Value Adjustment Factor, 이하 VAF)는 측정되는 애플리케이션의 일반적 기능에 등급을 부여한 14개의 일반시스템특성(General System Characteristics, 이하 GSC)에 기반을 두고 있으며 각 특성은 0 ~ 5까지의 6 등급으로 표현된다. VAF는 미조정기능점수의 규모를 -35% ~ +35%까지 조정하는 데 목적이 있으며 결정 절차는 다음과 같다.

① Step 1 : 14개의 GSC 각각에 대하여 영향도 점수(0 ~ 5점)를 부여
② Step 2 : 14개의 영향도를 합하여 총영향도를 산출
③ Step 3 : 아래의 방정식에 대입하여 조정인자 산출

VAF = (총영향도 * 0.01) + 0.65

VAF는 UFP(Unadjusted Function Point)에 곱해진 후 AFP(Adjusted Function Point)를 산출한다.

(Adjusted)FP = UFP * VAF

총영향도는 GSC의 총 합으로 이루어지며 14가지의 개별 특성과 각 특성별 영향도, 그리고 영향도 별 점수부과 표는 각각 <표 5>, <표 6>와 같다.

(표 5) 일반시스템 특성 및 영향도

번호	일반시스템특성	영향도
1	Data Communications	0 ~ 5
2	Distributed Data Processing	0 ~ 5
3	Performance	0 ~ 5
4	Heavily Used Configuration	0 ~ 5
5	Transaction Rate	0 ~ 5
6	Online Data Entry	0 ~ 5
7	End-User Efficiency	0 ~ 5
8	Online Update	0 ~ 5
9	Complex Processing	0 ~ 5
10	Reusability	0 ~ 5
11	Installation Ease	0 ~ 5
12	Operational Ease	0 ~ 5
13	Multiple Sites	0 ~ 5
14	Facilitate Change	0 ~ 5

(표 6) 시스템 영향도 점수부과 표

점수	시스템 영향도
0	해당사항 없음 또는 영향도 없음
1	주요하지 않은 영향
2	보통의 영향
3	평균적인 영향
4	주요한 영향
5	강한 영향

VAF는 시스템 고유의 특성을 규모에 반영할 수 있도록 Albrecht에 의하여 고안되었다. 하지만 발표 이후 다음과 같은 논란이 끊임없이 제기되어왔다[11].

① 영향도 등급 판정은 Albrecht의 주관적 관점이다.
② 모든 시스템특성을 평균적인 영향으로 산정하여도 +7%가 조정된다.
③ 산정하는 과정에서 비율(ratio)과 산술(ordinal)을 혼용하였다.

또한 14가지 특성이 현재의 다양하고 동적인 소프트웨어 환경을 반영하지 못하고 있는 것도 문제점으로 부각되고 있다[12].

3.3 복잡도 가중치

Albrecht의 FP 방식이 발표된 후 기능별 특성을 충분히 반영하지 못한 획일적 가중치는 논란의 대상이 되어왔다[11]. 발표 당시 복잡도에 의한 가중치는 'debate and trial' 방식으로 Albrecht에 의해 결정되었으며 IBM 조직의 특성이 반영된 것으로 이후 이를 개선하려는 많은 연구가 진행되어져 왔다[11].

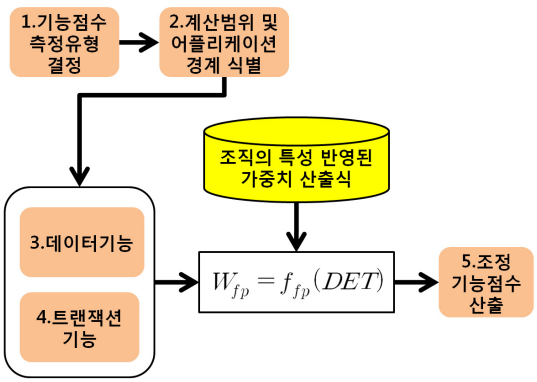
각 기능 별 가중치가 single matrix에 의해 정해지는 현재의 FPA 방식으로는 다양한 기능에 대해 차별적인 가중치를 부여하는 것이 불가능하다. 이는 공학용 계산이나 복잡한 알고리즘이 탑재된 소프트웨어의 규모를 측정하는데 있어 단점으로 작용한다.

(표 7) 기능점수 복잡도에 따른 가중치 부과표

참조파일 유형의 개수	복잡도		
	낮음	보통	높음
ILF	x 7	x 10	x 15
EIF	x 5	x 7	x 10
EI/EQ	x 3	x 4	x 6
EO	x 4	x 5	x 7

4. 개선된 FPA 모델

기존 FP 방식의 문제점을 해결하기 위하여 그림 4와 같은 개선된 기능점수 모형을 제안한다.



(그림 4) 개선된 FPA 모델

제안하는 모델에서는 아래와 같은 원칙을 적용하였다.

① 내부 복잡도 계산의 단순화
② 일반시스템특성 생략
③ 가중치 수치모델 제시

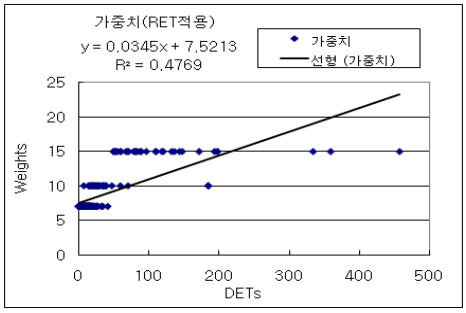
4.1 내부 복잡도 계산의 단순화

데이터 기능을 측정하는 과정에서 많은 노력이 들어가고 측정자 주관적 의견이 개입될 여지가 많은 부분이 RET 측정이다. 하지만 RET는 그 결정력이 DET가 갖는 결정력보다 오히려 작으며 [12], 기능점수 측정 매뉴얼에서도 RET와 DET의 식별은 논리 파일의 복잡도에만 영향을 미치기 때문에 데이터 기능 식별 자체에 더 많은 노력을 기울일 것을 강조하고 있다[5]. 이에 내부 조직 데이터를 활용하여 RET를 고려하여 데이터 기능점수를 측정한 경우와 RET를 고려하지 않고 DET만을 고려한 데이터 기능점수를 산정하여 비교한 후 다음과 같은 결론을 얻었다. (DET의 범위는 기존 FP 방식과 같으며 각 구간 별 복잡도는 '낮음', '보통', '높음'으로 가정하였다.)

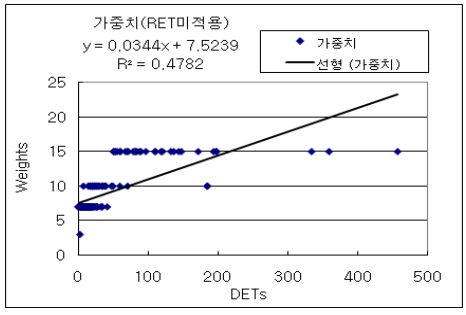
(표 8) 데이터 기능점수에 대한 RET의 영향력

시스템	데이터 기능점수	
	RET/DET 고려	DET만 고려
A	150	153
B	102	105
C	56	59
D	183	178
E	44	44
F	337	337
G	653	653
H	91	91
I	80	76
J	112	112

위 결론에서 보는 바와 같이 대부분의 시스템에서 RET를 고려하여 데이터 기능점수를 산정한 결과와 RET를 고려하지 않고 데이터 기능점수를 산정한 결과의 차이가 미미하였다. 이에 제안하는 모델에서는 측정자 주관적이고 투입 노력에 비하여 전체 기능점수에 미치는 영향력이 미미한 RET 산정 절차를 생략한다.



(그림 5) 기능점수 가중치에 대한 DET 결정력 (RET 적용)



(그림 6) 기능점수 가중치에 대한 DET 결정력 (RET 적용하지 않음)

4.2 조정인자

미조정기능점수가 조정된 기능점수보다 궁극적인 규모에 더 강하게 상호 연관되어 있다는 것이 발표되었으며[13] ISO/IEC 20926:2003 표준 역시 조정되지 않은 기능 점수를 기반으로 한다. 이에 조정인자 산출 과정을 생략하는 기능점수 산정 모델을 가정할 수 있으며 이를 통해 다음과 같은 이점을 얻을 수 있다.

① GSC와 관련된 논란 제거
② 측정 시간 단축

4.3 가중치 수치모델

데이터 기능 측정 절차를 단순화하고 일반시스템 특성을 생략한 모형의 알고리즘을 다음과 같이 가정할 수 있다.

(표 9) 가중치 산출 알고리즘

① 가중치 함수

$$W_{afp} = f_{afp}(DET) \text{ where } = W_{afp}\text{가중치}$$

② 추세식

$$W_{afp} = ax + b \text{ , where } a, b = \text{선형인자}$$

여기서 a와 b는 조직 내의 축적된 데이터를 통해 결정되는 선형인자로서 해당 조직의 특성을 반영하여 가중치를 결정하게 된다. 위 알고리즘을 통한 기능점수 가중치는 다음과 같은 장점을 갖는다.

① Parametric 알고리즘을 적용함으로써 Albrecht의 주관적이고 IBM 내부 조직 특성이 반영된 가중치 논란 제거
② 조직의 데이터가 쌓여갈 수록 해당 조직의 특성을 반영
③ RET 및 GSC 측정 절차를 생략함으로써 측정자 주관적 관점에서 오는 논란 및 측정 효율성 제고

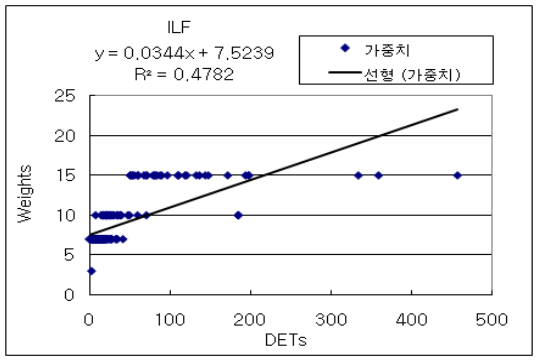
4.3.1 가중치 수치모델 계수 결정

조직 내에서 수집된 데이터를 기초로 위의 수치 모델을 적용하여 도출된 선형인자는 다음과 같다.

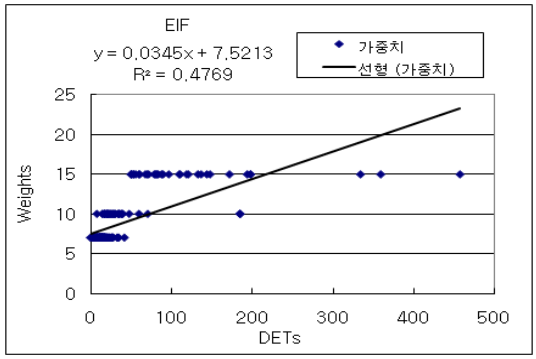
(표 10) 기능 유형 별 선형 인자

기능 유형	a	b
ILF	0.0344	7.5239
EIF	0.0345	7.5213
EI	0.0334	3.7495
EO	0.0116	5.0589
EQ	0.0128	3.7537

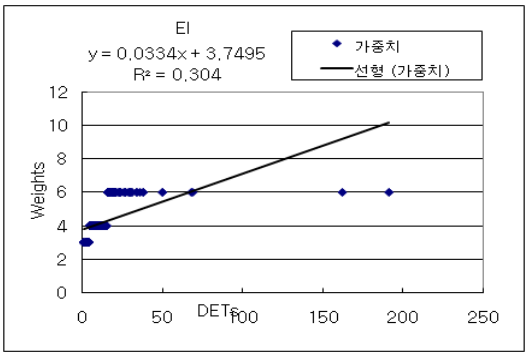
각 기능 타입 별 선형인자가 적용된 선형 추세 결과는 다음과 같다.



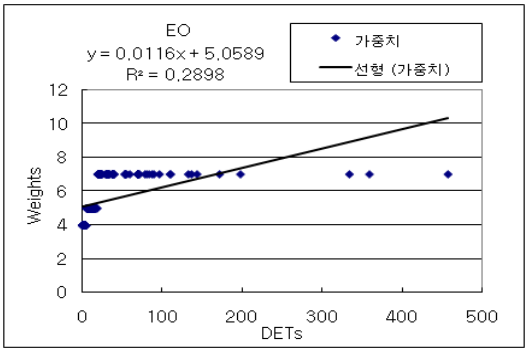
(그림 7) ILF 추세식 및 선형 인자



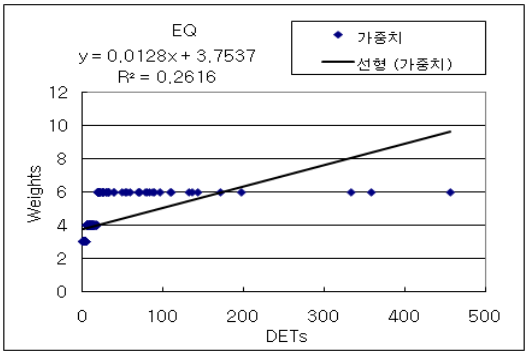
(그림 8) EIF 추세식 및 선형 인자



(그림 9) EI 추세식 및 선형 인자



(그림 10) EO 추세식 및 선형 인자



(그림 11) EQ 추세식

5. 검증 및 평가

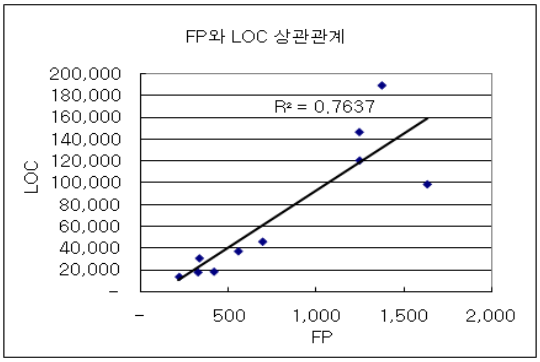
검증을 위하여 동일한 업무 도메인에 속해있는 10개의 애플리케이션을 대상으로 FP 전문가를 통해 기능 점수를 측정하였다.

개선된 모델로 측정한 기능점수와 IFPUG의 FP 방식으로 측정한 기능점수는 각 방식의 프로그램 LOC와의 상관관계를 비교함으로써 적합성을 평가하였다.

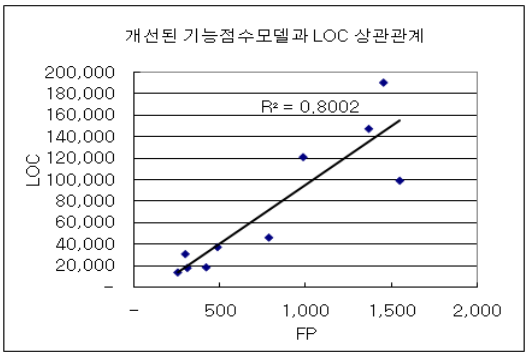
(표 11) FPA 방식과 개선된 기능점수모델의 LOC와의 상관관계

시스템	LOC	FPA	개선된모델
A	46,115	694	782
B	18,552	416	416
C	13,869	217	250
D	120,644	1,248	982
E	30,837	332	294
F	98,778	1,635	1,546
G	189,554	1,375	1,451
H	146,786	1,246	1,365
I	18,023	324	306
J	37,264	555	482

위의 결과로부터 기존 FP 방식과 LOC의 상관관계와 개선된 기능점수 모델을 사용하여 측정한 기능점수 결과와 LOC간의 상관관계를 구하였다.



(그림 12) FP와 LOC 상관관계



(그림 13) 개선된 기능점수 모델과 LOC 상관관계

위 결과로부터 본 논문에서 제안하는 개선된 기능점수 방식이 기존 FPA 방식보다 LOC와의 상관관계가 높은 것으로 나타났다. 다만 제한된 조직의 데이터를 통한 결과이므로 본 논문에서는 제안하는 모델의 성능 평가에만 그 의의를 두고자 한다.

6. 결론 및 향후 과제

최근의 소프트웨어 견적 방식은 기존의 물리적 코드나 경험에 의하여 비용이나 공수를 산정하는 방식에서 사용자가 이해하고 제공받는 기능을 중심으로 소프트웨어의 규모를 산정하는 FSM 방식으로 패러다임의 변환이 가속화되고 있다. 우리나라에서는 2004년 FSM 방식 중 하나인 FPA에 기반한 기능점수 방식을 도입하였으며 공공 및 대형 프로젝트를 중심으로 그 활용도를 넓혀가고 있다. FPA를 통한 소프트웨어 견적은 구현기술, 언어, 플랫폼과 독립적으로 규모를 산정할 수 있으며 사용자 언어로 요청된 사항을 규모로 산정하기 때문에 프로젝트 초기에 비교적 구체적인 요구사항을 도출할 수 있다는 것과 널리 쓰이는 국제 표준을 준용한 측정 방식이기 때문에 국가간 소프트웨어 경쟁력 비교가 가능하다는 것이 장점으로 부각되고 있다. 이러한 장점에도 사용자 관점을 강조한 결과로 존재하는 측정기준의 모호성과 1979년 Albrecht가 debate and trial 방식으로 결정한 주관적 복잡도 매트릭스는 끊임없는 논란의 대상이 되어왔다.

이에 본 논문에서는 기존에 제시되어온 문제점을 분석하여 새로운 기능점수 측정 모델을 제시하였다. 제시한 모델은 측정 과정에서 야기되는 모호성을 줄이고 측정 절차를 간소화하였다. 또한 다양한 조직의 특성과 기능의 복잡도를 반영할 수 있는 가중치 수치 모델을 적용하였다. 제시한 모델은 평가 결과 기존의 FPA 방식을 대체할 수 있음이 입증되었으며 특히 측정 절차의 간소화로 빠른 시간에 규모를 추정하여야 하는 제안 단계에서 그 활용도가 높을 것으로 기대된다.

기능점수 방식이 정착되고 활용이 확대되기 위해서는 다양한 소프트웨어 환경을 수용할 수 있는 모델을 개발하는 것이 시급하다. 특히 생산성에 영향을 미치는 개인의 능력이나 소프트웨어의 품질에 영향을 미치는 요소, 개발 방법론에 따른 차등적 보정 요소 등을 반영한다면 보다 신뢰성 있게 소프트웨어 규모를 측정할 수 있으리라 생각한다. 또한 국가 차원에서 다양한 산업 도메인별 측정 데이터를 수집하고 관리하는 노력이 필요하다. 수집된 데이터는 기능점수 당 단가 산정뿐만 아니라 생산성 분석 및 향후 소프트웨어 정책을 수립하고 계획하는 데 측정 지표로 삼을 수 있다. 이러한 노력은 우리나라의 소프트웨어의 생산성을 향상시키고 고부가가치 산업으로 거듭나는 밑거름이 될 것이다.

참 고 문 헌

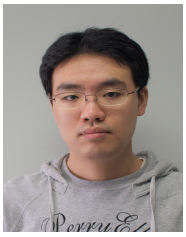
[1] Albrecht, A.J, 'Measuring Application Development Productivity', Proceedings of Joint SHARE, GUIDE, and IBM Application Development Symposium, pp.83-92, 1979.

[2] '소프트웨어 사업대가 기준', 정보통신부 고시 제 2004-8호, 2004.

[3] IBM CIS & a Guidelines 313, 'AD/M Productivity Measurement and Estimate Validation', 1984.

[4] http://www.ifpug.org
[5] IFPUG, 'Function Point Counting Practices Manual R4.2.1', 2004.
[6] http://www.nesma.org
[7] Symons, Charles, 'Function Point Analysis-Difficulties and Improvements', IEEE Transactions on Software Engineering, vol. 14, no. 1, pp.2-11, 1988.
[8] 조은숙, 'PFP 모델 : 소프트웨어 규모산정을 위한 개선된 Function Point 모델', 고려대학교 학위논문, 2005.
[9] Goodwin, Paul, 'Comparing Mark II with IFPUG Function Points', Proceedings of International Function Point User Group, 1994.
[10] Kemerer, Chris F., 'An Empirical Validation of Software Cost Estimation Models', Communications of the ACM 30, 1987.
[11] Fenton, N.E., Pfleeger, S.L., 1997. Software Metrics. A Rigorous & Practical Approach, second ed. PWS, Boston.
[12] 허승현, 최은만, "동적 웹 어플리케이션의 특성을 반영한 조정 기능 점수 산정 방안", 한국정보과학회 가을 학술발표논문집, vol.31, No.2, 2004
[13] Goodwin, Paul, "Comparing Mark II with IFPUG Function Points", International Function Point User Group, Westerville, OH, Fall 1994 Proceedings.

◐ 저 자 소 개 ◑



정 인 용(Jung, In Yong)
2008년 고려대학교 전기전자전파공학부(공학사)
2008~현재 고려대학교 대학원 전자전기공학과(공학석사)
관심분야 : Software Estimation, 분산 시뮬레이션
E-mail : dekarno@korea.ac.kr

우 덕 제(Woo, Doug Je)
2003년 한림대학교 정보공학과(공학사)
2009년 고려대학교 공학대학원 전자컴퓨터학과(공학석사)
2003～현재 한전KDN
관심분야 : Software Estimation, CMMi
E-mail : woosaint@korea.ac.kr

박 진 형(Park, Jin Hyeong)
2007년 고려대학교 전기전자전파공학부(공학사)
2007~현재 고려대학교 대학원 전자전기공학과(공학석사)
관심분야 : 클러스터컴퓨팅, 영상인식
E-mail : kanonerin@korea.ac.kr

정 창 성(Jeong, Chang Sung)
1981년 서울대학교 전기공학과 졸업(공학사)
1984년 Northwestern University 전자계산학과 졸업(공학석사)
1987년 Northwestern University 전자계산학과 졸업(공학박사)
1987년～1992년 포항공과대학교 전자계산학과 조교수
1992년～1998년 고려대학교 전자공학과 부교수
1998년～현재 고려대학교 전자공학과 정교수
관심분야: Distributed & Parallel Supercomputing, Grid Computing, Ubiquitous Computing, Network Virtual Computing
E-mail : csjeong@korea.ac.kr