기능점수를 이용한 소프트웨어 규모 및 비용산정 방안에 관한 연구

김현수*

An Improvement of Software Sizing and Cost Estimation Model with Function Point Methods

Hyunsoo Kim*

ABSTRACT

Software cost estimation is an important issue both for buyers and sellers(developers). We reviewed domestic and foreign researches and practices on software cost estimation with function point method comprehensively. In this paper, we derived four promising alternative function point models. They are an IFPUG(International Function Point User Group)-based model(Model I), a shorthand model for client/sever software systems(Model II), a data-oriented model for relatively large software projects(Model III), and a general-purpose function point model for non business application softwares as well as business applications(Model IV).

Empirical data shows that Model I, II, and IV are very useful function point models. In particular, model II and IV look very useful models since they are concise and accurate. These models can be incorporated in a new improved guideline for software cost estimation. General opinion survey shows that Model I, II, and IV are preferrable. There are no significant differences in preference between buyers and sellers. The survey also shows that users think function point method is better than step(line of code)-oriented cost estimation methods in many ways including objectivity and estimation accuracy.

* 국민대학교 정보관리학과
(Department of Management Information Systems, Kookmin University)

1. 서 론

프로그램 라인수(Line of Code)로 소프트웨어의 규모와 개발기간을 산정하는 소프트웨어 비용 모형이 개발된 이후 산업계와 학계에서는 보다 정확한 소프트웨어 규모 견적이 중요한 과제가 되어왔다. 규모견적 방식의 최근 동향은 프로그램 라인수와 같은 개발자 관점의 물량중심 방식보다는 사용자 입장에서의 경제적가치를 고려하는 소프트웨어 규모 및 비용산정 방식이 보다 선호되면서 활발히 연구되고 있다. 사용자 중심의 가치를 기준으로 소프트웨어의 규모를 견적하는 대표적인 비용산정 모형이 기능점수(Function Point) 모형이다. 기능점수 모형은 1979년 Albrecht[7]에 의해 개발된 방식으로서, 정보처리 규모와 기술적 복잡도 요인에 의하여 소프트웨어의 규모를 산정하는 방식이다. 정보처리 규모는 사용자 입장에서 본 시스템의 기능을 외부입력, 외부출력, 내부논리파일, 외부인터페이스파일, 외부조회 등의 5가지 유형으로 나누어 각 기능의 복잡도를 고려하여 측정한다. 기술적 복잡도 요인은 데이터통신, 분산처리 등 14가지의 시스템 특성에 의해 계산하며, 이 값이 정보처리규모에서 구한 기능수치를 보정하여 최종 기능점수를 계산하게 된다. 기능점수는 개발자 입장에서의 소프트웨어 견적량인 소스코드의 크기와는 무관한 추상적인 값으로서 최종사용자 입장에서 소프트웨어 규모를 견적하는 값이 된다.

본 연구는 기능점수 모형의 본질적인 예측능력과 활용성을 제고하는 방안을 제시한다. 즉, 기존의 호스트 중심 개발환경에 적합하게 만들어진 기능점수 모형을 보완하여 클라이언트/서버 환경에도 적용할 수 있는 모형을 개발한다. 또한 사무처리 중심의 소프트웨어 개발에 적용할 목적으로 개발된 원래 기능점수 모형의 활용범위를 넓히기 위한 여러가지 대안을 제시하고 실데이터를 이용하여 이를 검증한다. 이러한 과정을 통하여 소프트웨어 개발비용에 대한 예측력과 활용의 용이성을 함께 만족시키는 바람직한 한국적 모형을 도출하는 것이 본 연구의 주된 목적이다. 본 연구는 문헌 연구와 사용자의 기능점수 방법에 대한 의견 수집 및 실제 프로젝트 데이터 수집에 의한 통계분석 등을 통하여 수행되었다.

제 2 장에서는 기능점수 모형의 개요를 소개하고, 그동안의 관련 연구 결과를 포괄적인 관점에서 정리하여 제시한다. 제 3 장에서는 기존 모형의 문제점을 해결하는 복수개의 개선모형을 제시한다. 우리나라의 실제 프로젝트 데이터와 소프트웨어 업계 전문가의 의견을 수집하여 개선 모형의 타당성을 분석한 결과를 제 4 장에서 제시하고 토의를 수행한다. 마지막으로 제 5 장에서는 연구결과를 요약하고 향후 연구방향을 제시한다.

2. 기능점수 모형 연구

2.1 기능점수 모형 개요

기능점수는 기능수 계산, 기술적 복잡도 계산, 기능점수 계산 등의 3단계 절차에 의해 계산된다. 계산과정을 요약하면 다음과 같다.

가. 기능 수(Function Count: FC) 계산

기능점수 모형에서는 정보처리규모를 기능 수(FC)로 계산한다. 기능수는 개발 대상 소프트웨어가 명확하게 정의되고, 외부 시스템과의 인터페이스가 분명해진 상태에서 다음과 같이 5가지 유형의 기능을 추출하여 각각의 복잡도를 측정한다.

① 외부입력 (external input types): 시스템(소

〈표 2-1〉 기능유형별 가중치

기능 유형	가 중 치			기능 수 (FC)
	단 순	보 통	복 잡	
외부 입력	_ × 3 = _	_ × 4 = _	_ × 6 = _	
외부 출력	_ × 4 = _	_ × 5 = _	_ × 7 = _	
내부논리파일	_ × 7 = _	_ × 10 = _	_ × 15 = _	
외부인터페이스파일	_ × 5 = _	_ × 7 = _	_ × 10 = _	
외부조회	_ × 3 = _	_ × 4 = _	_ × 6 = _	
기능수(FC) 합계				

프트웨어시스템을 의미함)의 경계 밖에서 시스템에 데이터를 입력하는 기능.

② 외부출력 (external output types): 시스템의 경계밖으로 데이터를 출력하는 기능.

③ 내부논리파일 (logical internal file types): 시스템내에서 논리적으로 유지되는 파일로서 사용자 관점에서 볼때 시스템에 의해 생성되고, 사용되며, 또한 관리되는 논리적인 데이터의 그룹.

④ 외부인터페이스파일 (external interface file types):. 응용시스템간에 공유되거나 전송되는 파일로서 시스템 밖의 응용시스템이 논리적으로 유지하는 파일.

⑤ 외부조회 (external inquiry types): 입력이 출력을 즉시 요구하는 입력/출력조합을 외부조회로 판단한다. 입력 또는 출력중의 하나라도 포맷(format)이 다르거나, 내부 처리로직이 다르면 별개의 기능으로 간주함.

각 기능의 복잡도는 단순, 보통, 복잡의 3 단계로 구분하여 평가한다. 복잡도에 주어지는 가중치는 기능 유형별로 상이하다. Albrecht는 많은 토론과 시행착오를 거쳐 〈표 2-1〉과 같은 가중치를 도출하였다.

위 표의 가중치는 비교적 간단한 복잡도 판단기준에 의해 결정되는 값이다. 이러한 단점을 보완하기 위하여 최근의 기능점수 모형은 기능에서 다루는 데이타 요소의 수, 입출력 기능에서 참조하는 화일수와 화일이 포함하는 레코드의 종류수의 조합 등 2차원 테이블에 의해 기능의 복잡도 가중치를 판단하고 있다. [3]

나. 기술적 복잡도(Technical Complexity) 계산

기능수(FC)는 기술적 복잡도를 이용하여 생산성을 측정하는 지표인 기능점수로 변환된다. 기술적복잡도는 아래와 같은 14개의 항목에 대해 영향도를 평가하여 계산한다. 영향도(Degree of Influence:DI)는 0 부터 5까지의 정수로 나타낸다.

14개의 기술적 복잡도 요소는 다음과 같다.

ⓐ 데이터 통신의 필요 정도

ⓑ 분산처리 기능의 정도

ⓒ 응답속도나 처리율(throughput)과 같은 시스템의 성능(performance) 요구

ⓓ 운용될 시스템의 여유정도 (사용정도)

ⓔ 트랜잭션율(Transaction rate)이 높은 정도

ⓕ 온라인 자료입력 및 제어기능 요구정도

⑧ 온라인입력이 복수개의 스크린 또는 오퍼레 이션을 사용하는 트랜잭션으로 구성되는 정도
⑨ 내부논리파일을 온라인 갱신하는 정도
⑩ 처리의 복잡성 정도 (예: 많은 제어 상호작용과 의사결정 수반, 광범위한 논리 및 수학식 사용, 많은 예외처리로 인한 불완전한 처리)
⑪ 프로그램 재사용의 고려 정도
⑫ 변환 및 설치 용이성을 고려하는 정도
⑬ 효과적인 백업, 복구 등 운용상의 편리함을 고려하는 정도
⑭ 복수의 조직을 위한 복수 장소에의 설치를 고려하는 정도
⑮ 변경 용이성을 고려하는 정도

14개 항목의 영향도를 평가하여 합산한 총영향도는 0 부터 70 사이의 값이 된다. 기술적 복잡도 값인 TCF(Technical Complexity Factor)는 다음 식으로 계산된다.

$$TCF = 0.65 + 0.01 \times \text{총영향도}$$

기능점수(FP)는 FC와 TCF로부터 다음과 같이 계산된다.

$$FP = FC \times TCF$$

TCF 값이 0.65 에서 1.35 사이의 값이므로, 기능점수는 기능수를 +/- 35% 범위내에서 보정하여 얻어지는 값이 된다. 기능점수(FP)는 생산성 및 품질, 비용의 기준으로 이용될 수 있다. 예를 들어 생산성 = FP / MM(man-month), 품질 = 결함(defects) / FP, 비용 = 원 / FP, 문서량= 문서페이지수 / FP 등으로 견적할 수 있다.

또한 투입인원 또는 스텝수와 FP와의 관계를 도출하여, 다음과 같이 관계식을 정립할 수 있다.

투입인원(MM) = $a \times (FP)^b$, (1), 또는

투입인원(MM) = $c \times (FP) + d$, (2), 또는

스텝수(LOC) = $e \times (FP) + f$ (3).

여기서 a, b, c, d, e, f 는 상수이다.

Albrecht와 Gaffney[8]는 COBOL과 PL/1을 사용한 24개 시스템에 대하여 (2)와 같은 모형을 사용하여 투입시간을 추정하였다. 작업시간(Work-Hours)으로 나타낸 회귀식은 다음과 같다.

소요 작업시간 (Work-Hours:인시) = $54 \times (FP) - 13{,}390$ ($R^2 = 0.7642$)

위 식은 기능점수가 248개 이상 (실제로는 이보다 큰 값 이상에서 성립)인 시스템에서 성립되는 식으로서, 기능점수와 소요공수는 선형적인 관계를 가짐을 나타내고 있다. 그러나 예측식의 적용범위가 상하한이 제한되어 있어, 선형성의 여부를 판단하기 위해서는 보다 정밀한 분석이 요구된다. 본 연구에서는 이러한 선형성 가정에 대해서도 분석을 수행하였다.

Mendelson[16]은 Albrecht와 Gaffney[8]등이 조사한 39개 프로젝트 자료를 이용하여 (1)식을 사용하여 소요공수 예측식을 다음과 같이 도출하였다.

소요공수(MM) = $0.0374 \times (FP)^{1.23}$ ($R^2 = 0.7000$)

위 식은 스텝수 기준 소요공수 예측방식인 COCOMO 모형과 유사하며, 전형적인 규모의 비경제(즉, 규모가 증가하는 비율 이상으로 소요공수가 증가함)를 나타낸다.

2.2 기능점수 모형의 문제점 및 개선에 대한 연구

기능점수 모형은 사용자 관점에서 소프트웨어를 견적하는 유용한 방법이지만, 적용범위, 측정의 신뢰성, 가중치 적용 등에서 여러가지 문제점을 내포하고 있다. 기능점수 모형을 개선하는 연구는 모형의 구조를 개선하는 방향, 활용성을 향상하는 방안, 적용범위를 확대하는 방향등 3가지 방향으로 진행되고 있다. 각 방향에 대한 주요 연

구결과는 다음과 같다.

가. 모형 구조 개선 연구

모형 구조의 개선 방향으로 대표적인 연구는 Symons[20]와 일본 COSDES(Committee On Software Development Estimation System)의 연구[1, 3, 6]이다. Symons는 Albrecht의 기능점수 모형이 기능의 복잡도를 너무 단순화하고 있으며, 기능의 가중치가 많은 토론과 시행착오를 거쳐 결정되었지만, 여전히 많은 의문이 제기되고 있다고 지적한다. 또한 내부처리의 복잡도가 기능분류에서 반영되지 않고, 기술적 복잡도의 14항목중의 하나로만 반영이 되어있기 때문에, 그 영향도를 충실하게 반영할 수 없으며, 기능점수는 '합산이 가능한' 점수가 아니다. 기술적 복잡도의 14가지 요소는 서로 중복되는 항목이 있다. 예를 들어, 시스템의 여유정도, 트랜잭션율(Transaction rate) 등과 응용시스템의 성능은 아주 밀접한 관계에 있는데 별도의 항목으로 계산되고 있다. 이와 같은 문제점을 개선하기 위하여 제안된 Mark II 방법은 트랜잭션(Transaction)의 입력, 처리(Process), 출력을 중심으로 기능 수를 측정한다. 이중에서 처리의 측정을 직접하기 어려우므로, 처리를 하기 위해 참조되는 데이터파일(요구분석 및 외부설계 단계에서는 엔티티에 해당하므로 참조되는 엔티티 타입을 측정함) 수와 관련하여 처리의 정도를 추정하는 모형을 사용하고 있다.

Mark II에서는 기능수를 조정하기 위하여 기존의 14개 복잡도 요소에 다른 응용시스템과의 인터페이스 정도, 특별하게 요구되는 보안기능 정도, 제 3자의 직접 접근 요구 정도, 문서화에 대한 요구정도, 교육훈련 하부시스템 등과 같이 특별하게 요구되는 사용자 교육훈련 요구정도, 사용자가 특별히 요구하는 하드웨어나 소프트웨어를 정의, 선택, 설치하는 요구정도등 6개의 요소를 추가하여 20 개의 요소를 사용하고 있다. Symons의 연구에 의하면 각 요인이 미치는 영향도가 Albrecht가 제시하였던 0.01 보다 작은 0.005 정도가 되는 것으로 분석되었다. 그 이유는 기술의 발전으로 구현이 보다 쉬워지는 경향을 반영하는 것으로 해석할 수 있다. (기능점수 모형이 완전히 기술에 독립적인(technology-independent)한 방법은 아님을 알 수 있다.) 문서화에 대한 요구정도는 상대적으로 만족시키기에 많은 노력이 드는 것으로 분석되며, 설치용이성, 운용용이성, 보안요구 등은 상대적으로 구현하기 쉬운 것으로 분석되었다.

일본 COSDES의 기능점수모형은 보다 많은 구조 변형을 시도한 모형이다. COSDES에서 사용하는 기능 유형은 화면, 대장(출력보고서), 화일로서 화면은 다시 입력, 출력, 입출력, 메뉴로 세분되고, 화일은 입력, 출력, 입출력으로 세분하여 기능점수를 계산한다. COSDES회원들의 과거 개발실적을 이용하여 모형의 신뢰성을 평가한 결과 오차율은 최대 +/- 40 % 정도였으며, 규모가 10,000 스텝 미만의 프로젝트는 오차가 큰 경우가 많았고, 규모가 20,000 - 100,000 스텝 정도의 개발 프로젝트는 오차가 +/- 20% 정도로 나타났다. 대상 시스템의 대부분은 사무분야의 온라인 시스템이었으며, 개발언어는 COBOL 또는 PL/I 이었다. [3]

개별 회사단위에서는 간이기능점수법을 개발하여 사용하고 있다. 대표적인 사례가 일본의 동양정보시스템, NEC, 후지쯔 등의 방법이다. 현재 동양정보시스템(TIS)이 제조업을 위주로한 클라이언트/서버 시스템 견적에 간이기능점수법을 사용하고 있다. 이 방법은 온라인과 배치로 처리를 구분하여, 온라인은 온라인입력(Entry), 조회, 장표(출력)으로 구분하며, 배치는 처리와 장표로 구

분하여 소프트웨어 규모를 견적한다. 또한 DB 엑세스 수를 조회와 갱신으로 구분하여 계산하고 난이도의 결정에 이용한다. NEC와 후지쯔 등은 TIS보다 간편한 기능점수모형을 보급하여 전사적으로 클라이언트/서버 시스템 견적에 사용을 권장하고 있다. NEC는 화면 1매, 장표 1매당 개발공수를 전형적인 '도구(tool)+미들웨어(middleware)+관계형데이터베이스(relational database)'의 조합에 대해 설정하고 있다. 조합의 종류는 'Excel+SequeLink+Oracle', 'uniFace+SQL*Net+Oracle' 등 7가지가 있다. 후지쯔는 시스템의 규모를 화면수만으로 견적한다. 데이터베이스 설계와 장표작성 부분은 종래의 시스템과 동일한 방법으로 견적한다. Visual Basic, Power Builder 등 8가지의 어플리케이션 개발도구, Excel 등 4 종류의 스프레드시트, Access등 6종류의 PC용 데이터베이스 시스템을 대상으로 개발 규모를 견적한다. [6]

간이기능점수법과 원래의 기능점수법의 단점을 개선하기 위하여 몇가지의 개량 기능점수법이 고안되고 있다. 즉 간이기능점수법보다 정밀도가 높고, 본래의 기능점수법보다 사용하기 간편한 방법이 개량기능점수법인데, 일본IBM과 CSK에서 사용하는 데이타 항목수에 의한 견적법이 대표적인 사례이다. 일본IBM에서는 SI 사업의 견적을 위해 자료사전에서 나타나는 데이터 항목수를 기준으로 소프트웨어의 규모를 견적한다. 일본IBM은 3,000건에 이르는 개발 사례로부터 축적된 데이터를 바탕으로 모형을 구축하였으며, SI 사업 견적에만 사용하고, 사내용 시스템 견적에는 개발도구와 기종(주로 범용기종 사용)이 안정되어 있기 때문에 기존의 기능점수 모형을 사용하고 있다. 클라이언트/서버 시스템은 하드웨어의 수나 종류, 개발도구, 미들웨어, 데이터베이스 등을 자유롭게 조합시킬수 있어 자유도가 매우 높아진다. 따라서, 시스템을 여러개의 패턴으로 나누어 각 패턴마다 최적의 견적방법을 개발하는것이 바람직한데, 패턴의 구분 기준으로는 시스템 구조(서버의 종류, 클라이언트의 종류, C/S 시스템의 계층구조 등), 소프트웨어(OS, 데이터베이스, 클라이언트 개발 도구 등), 개발방법(프로그래밍 언어 중심, 또는 패키지 조합 중심 등) 등 3가지 항목에 기초하는 방안을 모색하고 있다. 예를 들어, 'UNIX 서버가 1대, Windows 클라이언트가 수십대, Oracle 과 Visual Basic사용' 등으로 하나의 패턴을 정의할 수 있다. 히다찌는 최근에 소프트웨어 개발방법론 HIPACE에 RAD를 기초로한 나선형방법(spiral approach) 개발 표준 수순(단계)를 확립하고, Visual Basic등 특정의 tool에 대응하는 '간이개발형 표준수순', OMT(Object Modeling Technique)법을 기초로한 '객체지향 개발수순'을 추가하여 견적의 정밀도를 향상시키고 있다. [6]

나. 모형의 신뢰성 향상 연구

Kemerer와 Porter [14]는 기능점수 측정상의 주요 오차요인에 대한 경험적인 분석을 수행하여 비교적 소수의 요소만이 기능점수 측정치의 오류수준에 영향을 미친다고 분석하였다. 백업파일을 세는 방식, 에러메시지를 세는 방식 등 14가지의 질문을 통하여 3 Site에서 기능점수의 변화를 분석한 결과, 백업파일을 내부논리파일로 취급하는 경우에 기존의 기능점수가 평균 29.7% 증가된 값으로 측정되었다. 백업파일을 외부출력으로 취급하는 경우에는 기능점수값이 17.7% 증가되고, 추가/변경/삭제 트랜잭션을 외부출력으로 취급하는 경우에는 평균 2%가 감소되는 것으로 나타났다.

Low와 Jeffery[15]는 기능점수 측정에 대한 실험을 수행하여 다음과 같은 결과를 도출하였다. 한 조직내에서 기능점수를 사용할때, 측정치의 오

차는 30% 이내인 것으로 분석된다. 서로 다른 조직에서 기능점수를 사용할 경우, 조직간에 측정값의 오차가 생길 수 있다. 기능점수를 잘 적용하기 위해서는 적용 경험이 필수적이다. 소프트웨어 개발경험이 많은 사람의 경우 기능점수 활용 경험이 많은 사람과 비슷한 정도의 측정 정확성을 보이는 것으로 나타났다.

다. 모형의 적용 범위 확장 연구

원래 기능점수 모형이 입출력을 중심으로 하는 사무처리시스템에 적용하는 목적으로 개발되었기 때문에, 적용범위에 대한 확장이 필요하다. 즉 입출력의 양은 많지 않지만, 내부에서 많은 복잡한 계산이 이루어지는 소프트웨어의 경우, 현재기준으로는 정확한 규모산정이 어렵다. 따라서 Jones[13]는 기능점수 모형을 변형하여 특성점수(Feature point)을 개발하고, 이를 시스템 및 엔지니어링용 소프트웨어의 견적에 적용할것을 제안하였다. 특성점수 모형은 내부처리 알고리즘이 복잡한 소프트웨어, 즉 실시간용 소프트웨어, 프로세스 제어용 소프트웨어, 장비내장형 소프트웨어 등의 규모를 견적하는데 유용하게 활용될 수 있다.

특성점수를 계산하는 방법은 기능점수를 계산하는 방식과 유사하다.

특성점수 = 특성수 × TCF

특성수를 계산하는 특성 유형은 기능유형에 알고리즘 항목을 추가한것이 특징이며, 점수의 가중치는 기능점수의 가중치중 '보통'에 해당하는 가중치를 사용하였다. 내부논리파일의 '보통'에 대한 가중치값 10을 7과 3으로 분할하고, 3을 알고리즘에 할당하여 전체적으로 평균가중치 값이 기능점수와 동일하게 유지되도록 하고 있다. 특성점수와 기능점수는 사무처리시스템이나 일반적인 엔지니어링 소프트웨어에서는 동일한 값을 나타내지만, 복잡한 실시간 소프트웨어 등에서는 특성점수가 기능점수보다 25 - 30 % 정도 높은 값으로 나타나고 있다. [18]

Mukhopadhyay와 Kekre[17]는 기능점수 모형을 프로세스 중심 시스템에 적용하기 위해 기능수(Function count: 조정되기전의 기능점수) 중심으로 소프트웨어 규모를 견적하는 모형을 개발하였다. 이들은 공장에서의 프로세스 제어 기능을 세가지로 나누고 있다. 프로세스의 커뮤니케이션 기능 (communication feature: CF), 위치를 제어하는 위치기능(position feature:PF), 동작을 제어하는 동작기능(motion feature:MF) 으로 기능을 구분하여 기능수를 산정한다. 소요공수는 기능수(FC)와 개발자능력(programmer speed: PS), 개발일정의 압박(schedule pressure: SP) 을 이용하여 계산된다. 이 모형의 상대오류((실제소요공수-추정소요공수)/실제소요공수)는 21% 정도로서 기능점수 모형의 상대오류 61% 와 비교하여 약 1/3 정도로서 추정의 신뢰도가 높은것으로 나타났다.

3. 기능점수 개선 모형

본 장에서는 앞서의 이론적 연구를 토대로 기능점수 모형의 실용성을 검증하고 우리나라의 개발 환경에 맞는 개선 모형을 도출하기 위하여, 복수개의 바람직한 대안 모형을 제시한다. 본 연구에서는 전통적인 기능 점수 모형의 단점을 개선한 대안을 먼저 제시한다. 이 모형은 호스트중심 환경에서 운용되는 소프트웨어는 물론이고 모든 범용 소프트웨어의 규모 견적에 사용될 수 있는 모형으로 개발된다. 또한 클라이언트/서버 시스템 견적을 비롯하여 최근에 급속히 변화되고 있는 소프트웨어 개발형태에 대하여 시스템 견적을 간

단하게 수행할 수 있는 간편모형을 개발하고 유용성을 검증한다. 간편모형의 또 다른 유형으로서 SI 사업등 대규모 시스템 견적을 위주로한 방법인 데이터 중심 모형의 타당성을 분석하고, 마지막으로 중대형 시스템을 포함한 모든 종류의 시스템에 공통으로 적용될 수 있는 개선된 기능점수 모형을 제시하고 실데이터로 검증한다. 이들 모형의 상세 구조와 내용은 아래와 같다.

3.1 모형 I

본 모형은 현재 가장 많은 사용자 집단을 가지고 있는 IFPUG(International Function Point User Group: 국제적인 기능점수 사용자 그룹)의 모형을 기본 골격으로 삼아, 우리나라의 산업 및 개발관행에 맞는 모형으로 변형하여 구축한다. 국제적으로 공인되고 있는 (1993년 12월 ISO(International Standard Organization)의 정식 검토 항목으로 인정되었음) 모형을 우리나라의 실제 프로젝트 데이터로서 최초로 검증하고, 우리에게 보다 적합한 모형으로 탄생시키기 위하여 튜닝(Tuning)하는 목적으로 본 모형을 구축한다.

가. 모형의 정의 및 체계

본 모형은 IFPUG의 기능점수 모형에서 사용하는 5가지 시스템의 기능을 골격으로 사용한다. 즉, 사용자 입장에서 본 시스템의 기능으로서 외부입력 (external input types), 외부출력 (external output types), 내부논리파일 (logical internal file types), 외부인터페이스파일 (external interface file types), 외부조회 (external inquiry types) 등의 5가지 유형으로 나누어 각 기능의 복잡도를 고려하여 측정한다.

기술적 복잡도 요인은 그동안 수행된 기능점수 모형의 개선방안에 대한 연구 결과를 반영하여 개선한다. 기존의 +/- 35 % 의 총 영향도 골격은 유지하되, 하드웨어 및 소프트웨어 환경의 변화를 반영하여 시스템의 특성 요인을 조정한다. 기술적 복잡도 요소는 변화된 시스템 개발 환경을 반영하여 12개의 복잡도 항목을 사용한다.

12 개의 항목은 기존의 IFPUG 모형에서 복잡도 요소에 포함되었던 '트랜잭션율(Transaction rate)이 높은 정도' 와 '변환 및 설치 용이성을 고려하는 정도'를 삭제하여 작성된 것이며, 'ⓚ 처리의 복잡성 정도', 'ⓛ 복수의 조직을 위한 복수 장소에의 설치를 고려하는 정도'에는 다른 항목에 비해 2배의 가중치를 부여하였다. '변환 및 설치 용이성을 고려하는 정도'를 삭제한 이유는 이 요소가 사용자에게 직접적으로 관련있는 항목이 아니고 프로젝트 팀 관련 항목이기 때문에, 개발원가보다 경제적 가치를 주로 고려하는 기능점수 모형의 철학에 보다 충실하기 위하여 삭제하였다. 'Transaction rate가 높은 정도'를 삭제한 이유는 이 요소가 성능요소에 충실히 반영되고 있기 때문이다. 내부처리의 복잡도에 2배의 가중치를 부여한 이유는 여러 연구자들이 지적하듯이, 기존 기능점수 모형을 사무처리 중심 모형에서 일반 모형에 좀 더 가깝도록 수정하는 과정이며, '복수개의 장소에 설치하여 운용되는 정도'에 2배의 가중치를 부여한 이유는 이러한 시스템의 경우 단일 시스템보다 구현 노력이 훨씬 많이 드는 것이 일반적이기 때문이다.

12개 항목의 영향도를 평가하여 합산한 총영향도는 0 부터 70 사이의 값이 된다. 기술적 복잡도 값인 TCF(Technical Complexity Factor)는 다음 식으로 계산된다.

$$TCF = 0.65 + 0.01 \times \text{총영향도}$$

기능점수(FP)는 FC와 TCF로부터 다음과 같이 계산된다.

$$FP = FC \times TCF$$

TCF 값이 0.65 에서 1.35 사이의 값이므로, 본 모형의 기능점수는 현재의 기능점수 모형과 같이 기능수를 +/- 35% 범위내에서 보정하여 얻어지는 값이 된다.

나. 개발비용 계산

기능점수는 다음과 같이 지수 모형 또는 선형 모형을 이용하여 소요공수를 계산하는데 이용된다. 모형의 적합성에 대한 판단은 수집된 실데이터를 이용하여 통계적인 적합성 분석을 수행한 후 결정한다.

지수모형: 소요공수(MM) = $a \times (FP)^b$,
선형모형: 소요공수(MM) = $c + d \times FP$,
여기서 a, b, c, d 는 상수이다.

소요공수가 추정되면 일반적인 비용산정방식에 의거하여 개발비용을 계산할 수 있다.

3.2 모형 II

소프트웨어 개발 초기 단계에서 간단히 쓸 수 있으면서도 적당한 예측 정확성을 가지는 모형을 도출하기 위해 현재까지 많은 모형이 제시되어 왔다. 그러나 지금까지의 모형은 호스트(Host)컴퓨터 중심 시스템의 단순 스텝(Step)수 산정에 따른 소요공수의 산정에 지나지 않았기 때문에, 현재의 클라이언트/서버 시스템에는 부적절한 것으로 판명되어지고 있다. 여기에 제시되어지는 모형은 클라이언트/서버 시스템에 관한 비용 추정 문제의 해결을 위해 화면과 장표, 그리고 DB 개발공수(Entity type 수)를 사용하며, 개발초기에 간단히 쓸 수 있으며, 적당한 예측 정확성을 얻을 수 있는 모형으로 개발된다. 본 모형은 클라이언트/서버 시스템 견적에 국한하지 않고 일반적인 시스템 규모 견적에도 활용될 수 있는 형식으로 제안된다.

가. 모형의 정의 및 체계

본 모형은 화면과 출력보고서수, 그리고 DB개발공수를 사용하여 기능수(Function Count: FC)를 산출한다.

FC = a×화면수 + b×출력 보고서수 +
c×DB개발공수(Entity Type 수)

① 화면수 : 소프트웨어에서 사용되는 총 화면수를 의미하며, 다중윈도우하에서는 표현될 수 있는 총 화면수를 의미한다. (입력화면, 조회 입력화면, 조회 출력화면 등)
② 출력보고서수 : 소프트웨어에서 출력되는 모든 출력보고서의 수를 의미한다.
③ DB개발공수 : 내부처리에 의해 생성되는 화일의 수를 제외한 기본화일수를 의미한다. 계산의 편의를 위하여 ERD(Entity Relationship Diagram)에서의 Entity Type 수로서 간접 측정한다.
④ 가중치 a, b, c : 가중치는 실 데이타에 대한 통계분석을 통하여 결정한다.

나. 개발비용 계산

기능수를 이용하여 직접 소요공수를 계산하고, 이에 기준하여 개발비용을 계산한다.

3.3 모형 III

일반적으로 정보시스템의 크기는 자료처리 정

도와 관계가 많다. 자료처리는 자료항목과 처리내용에 따라 결정되는데 자료처리 내용이 평균적으로 비슷한 수준이라고 가정할때, 자료항목수가 시스템 개발 규모를 측정하는 좋은 지표가 될 수 있다.

본 모형은 이와 같은 현재의 소프트웨어 규모 산정 모형에 기초하여 우리나라 상황에 가장 적합한 데이터 중심모형으로서 제안된다. 따라서, 우리나라의 소프트웨어 개발 환경과 인력구조에 맞는 모형을 도출하기 위하여 철저한 통계적인 검증작업을 수행한다.

본 모형에서는 요구사항정의가 끝난 시점에서 자료사전을 이용하여 다음과 같이 소프트웨어 규모를 산정한다. 자료사전이 생산되지 않는 방법론을 사용하는 경우, 자료항목을 파악할 수 있는 유사한 산출물을 이용하여 동일한 방법으로 견적한다.

① 자료사전의 자료항목을 중복을 허용하지 않고 계산한다. 사전에 나타나 있는 모든 항목을 카운트하되(선택이나 optional 도 각각 하나씩의 항목으로 계산), 같은 이름이 나타나는 경우 중복해서 카운트하지 않는다.

② 생산성통계를 이용하여 총 표준소요공수(Total SMM)를 계산한다.

총 표준소요공수(MM) = 자료항목수 × MM / 자료항목

이때, 자료항목의 수를 3단계로 구분하여 총 소요공수를 차별적으로 계산할 수 있다.

(예를 들어, 500 이하, 500-10,000사이, 10,000이상)

③ 본 연구에서는 다음과 같이 표준 소요공수(MM)를 종속변수로 하고, 자료항목수를 독립변수로 하여 소요공수 추정 회귀식을 도출한다.

표준 소요공수(MM) = a + b × 자료항목수

여기서, 프로젝트 형태에 따른 가중치를 사용하여 소요공수를 보정할 수 있다.

각 단계별 소요공수는 소프트웨어 개발 프로젝트 수행시에 수발주자간에 참고할 수 있는 기초자료로 우선 제시하며, 추후 상세모형의 도입을 위한 자료로 활용한다.

3.4 모형 IV

본 모형은 기능점수 모형의 추정 신뢰도와 사용 용이성을 동시에 만족시키기 위하여 개발되었다. 우리나라의 소프트웨어산업및 개발 관행에 맞는 모형이 되기 위해서는 기능수 계산이 보다 간편해야 한다. 업계에서 가장 중시하는 입력, 처리, 출력의 기본개념을 계량화하는 모형을 구축하여 개발 관행과 자연스럽게 부합되도록 한다. 처리부분의 계량화는 사무처리용 소프트웨어와 내부처리 알고리즘이 복잡한 소프트웨어의 특징을 모두 반영할 수 있도록 알고리즘수와 엔티티 타입수를 동시에 반영하여 모형을 구성한다. 기능점수를 계산할 때는 입력데이터의 유형수, 알고리즘의 수, 엔티티 타입의 수, 출력 데이터 유형수 등을 사용하며, 기술적 복잡도는 기존의 요소에 새로운 요소를 추가하여 구성하고, 기능점수를 계산한다.

가. 기능수(Function Count : FC) 계산

본 모형에서는 다음 식에 의해 기능수를 추정한다.

FC (조정하기 전의 기능점수)

$= a \times N_I + b \times N_P + c \times N_O$

여기서 N_I = 입력데이터 유형의 수

N_P = 처리 기능의 크기

N_O = 출력데이터 유형의 수

$N_P = v \times N_E + w \times N_A$

N_A = 알고리즘의 수

N_E = 참조되는 Entity Type 수

본 연구에서는 계산의 편의상 계수 b를 측정하지 않고, v, w, a, c 를 측정하는 4 변수 중회귀분석을 사용한다.

나. 기술적 복잡도(Technical Complexity) 계산

본 모형에서는 변화된 시스템 개발 환경을 반영하여 다음과 같은 17개의 복잡도 항목을 사용한다.

ⓐ 데이터 통신의 필요 정도
ⓑ 운용될 시스템의 여유정도 (사용정도)
ⓒ 온라인 자료입력 및 제어기능 요구정도
ⓓ 온라인입력이 복수개의 스크린 또는 오퍼레이션을 사용하는 트랜잭션으로 구성되는 정도
ⓔ 내부논리파일을 온라인 갱신하는 정도
ⓕ 프로그램 재사용의 고려 정도
ⓖ 효과적인 백업, 복구 등 운용상의 편리함을 고려하는 정도
ⓗ 변경 용이성을 고려하는 정도
ⓘ 분산처리 기능의 정도
ⓙ 다른 응용시스템과의 인터페이스 정도
ⓚ 특별하게 요구되는 보안기능 정도
ⓛ 제 3자의 직접 접근 요구 정도
ⓜ 문서화에 대한 요구정도
ⓝ 사용자가 특별히 요구하는 하드웨어나 소프트웨어를 정의, 선택, 설치하는 요구정도
ⓞ 처리의 복잡성 정도 (예: 많은 제어 상호작용과 의사결정 수반, 광범위한 논리 및 수학식 사용, 많은 예외처리로 인한 불완전한 처리)
ⓟ 복수의 조직을 위한 복수 장소에의 설치를 고려하는 정도
ⓠ 응답속도나 처리율(throughput)과 같은 시스템의 성능(performance) 요구

위의 항목 집합은 다음과 같은 기준에 의해 도출되었다. 즉, 기존의 IFPUG 모형에서 복잡도 요소에 포함되었던 '트랜잭션율(Transaction rate)이 높은 정도' 와 '변환 및 설치 용이성을 고려하는 정도'를 삭제하고, ' 다른 응용시스템과의 인터페이스 정도', ' 특별하게 요구되는 보안기능 및 제3자의 직접 접근 요구 정도', ' 문서화에 대한 요구정도' 가 추가되었으며, 기능점수 모형의 본질적인 단점을 보완하기 위하여 'ⓞ 처리의 복잡성 정도', 'ⓟ 복수의 조직을 위한 복수 장소에의 설치를 고려하는 정도', 'ⓠ 응답속도나 처리율과 같은 시스템 성능요구'에는 다른 항목에 비해 2배의 가중치를 부여하였다. 내부처리의 복잡도에 2배의 가중치를 부여한 이유는 기존 기능점수 모형을 사무처리 중심 모형에서 일반 모형에 좀 더 가깝도록 수정하는 과정이며, '복수개의 장소에 설치하여 운용되는 정도'에 2배의 가중치를 부여한 이유는 이러한 시스템의 경우 단일 시스템보다 구현 노력이 훨씬 많이 드는 것이 일반적이기 때문이다. 또한 '응답속도나 처리율(throughput)과 같은 시스템의 성능(performance) 요구'에 2배의 가중치를 부여한 것은 응답속도의 범주 안에 조회의 량과 그 처리에 관한 내용을 포함시킴으로서 클라이언트/서버 시스템에서 비중이 높은 조회를 처리하기 위해서이다.

기술적복잡도는 아래와 같은 ⓐ - ⓝ까지의 14개 항목에 대해서 앞서와 같이 영향도(Degree of

Influence:DI)를 0 부터 5까지의 정수로 나타낸다. 그리고, ⓞ, ⓟ, ⓠ 등 3항목은 0 부터 10까지의 정수로 영향도를 나타낸다. 17개 항목의 영향도를 평가하여 합산한 총영향도는 0 부터 100 사이의 값이 된다. 기술적 복잡도값인 TCF (Technical Complexity Factor)는 다음 식으로 계산된다.

$$TCF = 0.50 + 0.01 \times \text{총영향도}$$

다. 기능점수 및 개발비용 계산

앞에서 계산한 기능수와 복잡도에 대한 영향도를 이용하여 다음 공식으로 기능점수를 계산한다.

$$FP = FC \times TCF$$

TCF값이 0.5 에서 1.50사이의 값이므로, 본 모형의 기능점수는 +/- 50 % 범위 정도로 보정하여 얻어지는 값이 된다. 개발 비용 계산과정은 모형 I과 동일하다.

3.5 개발기간 추정

개발기간은 대부분 소요공수에 기준하여 추정된다. 대표적인 소요공수 모형으로는 COCOMO, Putnam 모형을 들 수 있다. Putnam의 모형[19]에서는 다음과 같은 구조의 모형이 사용되고 있다.

$$\text{개발기간(년)} = a \times \frac{(\text{프로그램라인수})^b}{(\text{소요공수})^c}$$

여기서 소요공수의 단위는 man-year이고, b = 3/4, c = 1/4 이다.

Boehm의 COCOMO 방식[11]에서는 아래와 같은 지수모형으로 개발기간을 추정하고 있다.

$$\text{개발기간 (DT: 월)} = a \times (\ MM\ )^b$$

COCOMO의 경우 a = 2.5 이고, b 는 프로젝트 유형에 따라, 0.32 (Embedded) - 0.38 (Semi-detached) 사이의 값을 가진다. 우리나라 기업중 일부에서는 b = 0.5 정도의 값을 사용하기도 한다. 이렇게 승수가 1보다 작은 이유는 프로젝트의 많은 활동을 병행작업으로 진행할 수 있고, 기간 증가에 따른 패널티(penality)요인이 매우 크기 때문이다.

이와 같이 승수가 1 보다 작은 지수모형의 타당성이 기존 연구에서 검증되고 있는 상황이므로, 본 연구에서도 지수모형을 사용하여 개발기간을 추정하는 모형을 제시한다.

4. 분석 결과 및 토의

본 연구는 2가지 방법으로 분석을 수행하였다. 우선 실제 프로젝트 데이터를 수집하여 통계적인 방법으로 모형의 타당성을 검증하였고, 이와 함께 국내 전문가의 의견을 수집하여 현업에서의 프로젝트 경험을 토대로 여러가지 측면에 대한 평가 의견을 수집하여 분석하였다.

의견조사설문은 개선 대안으로 제안된 복수개의 모형에 대한 여러가지 측면에서의 타당성 및 선호도 평가 항목을 조사하여 분석하였다.

프로젝트 실데이터 수집을 위한 설문의 대상은 소프트웨어 프로젝트 수주자 및 발주자의 역할별로, 또 프로젝트 규모별 및 형태별(사무처리 및 비사무처리)로 데이터를 수집하여 분석하였다.

설문지는 2차에 걸쳐 한국소프트웨어산업협회의 협조를 얻어 실데이터 설문 300 부, 의견조사설문 650 부를 배포하였다. 의견조사설문 응답은 99건 (응답율 15.2%), 실제 프로젝트 데이터 수집건수는 36건 이었으며 아래에서 분석 결과를 제시한다.

4.1 전문가 의견 분석

본 연구에서 개선 모형으로 제시한 4개의 모형에 대해 모형의 적합성, 활용성, 프로젝트 유형별 적합성에 대해서는 다음과 같은 결과를 얻었다.

소프트웨어 규모산정에 적용하기에 모형 IV가 가장 적합하다고 평가한 응답이 많았으며, 그 다음으로 모형 II, 모형 I 의 순으로 나타났다. 적합도에 대한 응답을 요약하면 다음 〈표4-1〉과 같다.

〈표 4 - 1〉 모형의 적합도 순위

(단위: %)

	모형 I	모형 II	모형 III	모형 IV
1 순위	20 %	22 %	14 %	33 %
2 순위	21 %	30 %	14 %	23 %
3 순위	27 %	27 %	24 %	10 %
4 순위	21 %	10 %	37 %	23 %
무응답	11 %	11 %	11 %	11 %
합계	100 %	100 %	100 %	100 %

활용성이 가장 높을 것이라고 응답된 모형은 화면수, 출력보고서수, Entity type수 등의 3가지 기능요소로만 견적을 하는 모형 II로 나타났다. 그 다음으로 모형 III, 모형 IV, 모형 I 이 비슷한 활용성을 가지는 것으로 나타났다.

예측의 정확성 측면에서는 모형 IV가 정확성이 가장 높을 것으로 응답되었다. 나머지 모형들은 거의 비슷한 수준의 정확성을 보일것으로 예상되었다.

프로젝트 유형을 사무처리, 비사무처리의 두가지로 구분할 경우, 프로젝트 유형을 가장 잘 설명할 수 있는 모형은 모형 IV, 모형 III, 모형 II, 모형 I 등의 순으로 응답되었다. 모형 IV가 다른 유형에 비해 매우 높은 순위로 응답된 이유는 '알고리즘수' 요인이 포함되었기 때문으로 분석된다. 즉, 알고리즘 수의 차이에 의해 사무처리와 비사무처리 소프트웨어의 견적이 뚜렷이 차이 날 수

〈표 4 - 2〉 모형의 활용성 순위

(단위: %)

	모형 I	모형 II	모형 III	모형 IV
1 순위	16 %	35 %	20 %	19 %
2 순위	13 %	30 %	23 %	22 %
3 순위	32 %	18 %	16 %	22 %
4 순위	28 %	6 %	30 %	26 %
무응답	11 %	11 %	11 %	11 %
합계	100 %	100 %	100 %	100 %

〈표 4 - 3〉 모형의 예측정확성 순위

(단위: %)

	모형 I	모형 II	모형 III	모형 IV
1 순위	20 %	23 %	15 %	28 %
2 순위	19 %	30 %	21 %	22 %
3 순위	30 %	25 %	16 %	13 %
4 순위	18 %	9 %	35 %	24 %
무응답	13 %	13 %	13 %	13 %
합계	100 %	100 %	100 %	100 %

있기 때문이다. 결과의 요약은 다음 〈표4-4〉에 있다.

각 프로젝트 유형별로 비용 견적에 가장 적합한 모형에 대한 의견을 질문한 결과 다음과 같은 응답을 얻었다. 사무처리용 소프트웨어 견적에는 모형 II가 가장 적합한 것으로 나타났으며, 과학기술용 소프트웨어의 견적에는 모형 I, 시스템용 소프트웨어의 견적에는 모형 I, 통신용 소프트웨

〈표 4 - 4〉 프로젝트 유형 설명력

(단위: %)

	모형 I	모형 II	모형 III	모형 IV
1 순위	16 %	16 %	19 %	33 %
2 순위	23 %	29 %	14 %	20 %
3 순위	25 %	23 %	23 %	14 %
4 순위	22 %	18 %	30 %	19 %
무응답	14 %	14 %	14 %	14 %
합계	100 %	100 %	100 %	100 %

〈표 4 - 5〉 프로젝트 유형별 비용 견적 적합성 순위

(단위: %)

	사무처리	과학기술	시스템	통신	공정제어	지휘통제	지능형, 멀티미디어
모형 I	10 %	32 %	32 %	33 %	31 %	23 %	19 %
모형 II	33 %	7 %	10 %	13 %	7 %	15 %	18 %
모형 III	10 %	15 %	12 %	12 %	13 %	18 %	11 %
모형 IV	26 %	21 %	22 %	18 %	26 %	20 %	27 %
무응답	21 %	25 %	24 %	24 %	23 %	24 %	25 %
합계	100 %	100 %	100 %	100 %	100 %	100 %	100 %

어의 견적에는 모형 I, 공정제어용 소프트웨어의 견적에는 모형 I, 지휘통제용 소프트웨어의 비용 견적에도 모형 I (무응답 제외), 지능형 및 멀티미디어용 소프트웨어에는 모형 IV가 가장 적합할 것으로 응답되었다.

소프트웨어 개발 프로젝트의 수주자와 발주자 간의 입장 차이로 인한 모형 선호도의 차이가 있는지를 분석한 결과, 전반적으로 발주자의 모형 선호도가 높은 것으로 나타나고 있으나, 상호간에 뚜렷한 견해 차이는 없는 것으로 나타났다.

4.2 실데이터를 이용한 계수 추정 및 모형 분석

본 연구를 위하여 수집된 36건의 실제 프로젝트 데이터를 이용한 계수 및 모형의 타당성 분석 결과는 다음과 같다.

가. 모형 I 에 대한 분석

표준 MM(정보통신부가 고시한 소프트웨어 개발비 산정기준의 생산성기준표를 이용하여 인건비 비례로 계산한 평균 MM)를 종속변수, IFPUG모형에 충실한 복잡도 요소로 계산된 기능점수(FP)를 독립변수로 하여 선형모형과 지수모형 각각에 대하여 분석하였다. 분석결과는 다음과 같다.

소요공수 (표준 MM) = 522 + 0.195 * FP
(R^2 = 0.5140, Prob>F: 0.0004)

소요공수 (표준 MM) = 6.95 $(FP)^{0.585}$
(R^2 = 0.3597, Prob>F: 0.0052)

Mendelson[16]의 분석에서는 지수모형의 승수 계수가 1.23 이었으며, 스텝수를 사용한 COCOMO에서도 1 이 넘는 결과가 도출된데 비해, 본 모형은 0.585 라는 1보다 작은 수치가 도출되었다. 이는 본 연구에서 수집된 프로젝트 데이터가 기업의 규모나 프로젝트 관행이 서로 다른 회사들에서 수집된 자료이기 때문에 측정치의 일관성이 부족하였기 때문인 것으로 분석된다. 또다른 이유로는 기능점수의 초기 연구들이 제시하는 모형과 같이 기능점수와 소요공수의 관계는 선형성이 강한 측면이 있다. 본 연구에서 선형모형의 설명력이 더 높게 나타나고 있는 것으로 보아, 이러한 선형성의 가설에 대하여 향후 보다 정밀한 분석이 필요할 것으로 생각된다.

나. 모형 II 에 대한 분석

여기서는 두 번의 회귀분석을 수행하였다. 먼저 보정되지 않은 기능점수인 기능수(FC)를 추정하는 회귀식을 도출하는 분석을 수행하고, 다음에 기능수를 이용하여 소요공수를 추정하는 모형을 도출하였다.

FC = - 849 + 3.52 * 화면수 + 22.96 * 출력 보고서수 + 14.40 * Entity type수
(R^2 = 0.7720, Adjusted-R^2 = 0.7150, Prob>F: 0.0004)

이 모형은 IFPUG 모형의 간편모형으로서 제시되었다. 즉 정밀하게 측정한 기능점수에 대응되는 간편한 척도를 개발하는 것이 목적이었다. 분석결과는 매우 희망적으로 제시되었다. 화면수, 출력보고서수, Entity type수 등 3 개의 변수만으로 5 개의 기능요인과 각각의 복잡한 난이도에 의해 측정된 기능점수를 대체할 수 있는 것으로 분석되었다. 이 모형은 또한 호스트 중심 시스템에서 운용되는 소프트웨어에 대한 견적뿐 아니라, 클라이언트/서버 시스템의 견적에도 잘 활용될 수 있는 방법으로서 현재의 기능점수 모형을 개선하여 프로젝트 실무에서 활용할 수 있는 좋은 대안이 될 수 있다.

기능수를 직접 독립변수로 이용하여 소요공수

(표준 MM) 추정 모형을 도출한 결과 다음과 같이 선형모형이 보다 유의한 것으로 나타났다.

소요공수 (표준 MM) = 433 + 0.216 * FC

(R^2 = 0.4920, Prob>F: 0.0001)

소요공수 (표준 MM) = 9.73 $(FC)^{0.543}$

(R^2 = 0.3362, Prob>F: 0.0024)

즉 모형 I 에서의 분석과 유사하게 도출되고 있으나, 복잡도에 의한 보정계수를 적용하여 보정된 기능점수(FP)를 사용하여 소요공수를 추정하는 것이 예측력이 높게됨을 보여주고 있다.

다. 모형 III 에 대한 분석

표준 MM를 종속변수, 자료항목수를 독립변수로 사용하여 단순회귀분석을 수행한 결과 지수모형과 선형모형 모두에 대해 전혀 유의한 결과를 얻지 못하였다. 선형모형의 R^2는 0.0052, 지수모형의 R^2는 0.0178 로 나타났고, 각각의 F value도 유의수준이 0.7207, 0.5065로서 유의도가 낮게 나타났다.

이러한 분석 결과로 유추할때, 자료항목수 하나만을 가지고 소요공수를 추정하기는 매우 어려운 것으로 판단된다. 그러나 자료항목수에 대한 보다 정밀한 자료수집과 함께 단계별 소요공수를 추정하는 등의 모형 변형과정을 거쳐 신뢰성있는 모형을 도출할 수도 있을 것이다. 이에 대한 추후 연구가 필요할 것으로 생각된다.

라. 모형 IV 에 대한 분석

여기에서는 모형 II의 분석과 유사하게 FC를 구하기 위한 모형의 계수를 추정하는 분석과 소요공수 추정 모형 분석을 수행한다.

FC를 추정하기 위해 입력데이터 유형의 수, Entity type수, 알고리즘수, 출력데이터 유형의 수 등 4 개 변수를 사용하여 도출된 모형은 다음과 같다.

FC = 692 + 0.426 * N_I - 1.165 * N_E + 6.739 * N_A + 6.232 * N_O

(R^2 = 0.9342, Adjusted-R^2 = 0.9078, Prob>F: 0.0001)

여기서 N_I = 입력데이터 유형의 수

N_E = Entity type 수

N_A = 알고리즘수

N_O = 출력데이터 유형의 수.

이 모형의 설명력은 상당히 높게 나타났다. 보다 정밀한 분석을 위해 각 변수의 개별적인 영향도를 분석하였다. 즉 각 변수를 독립변수로 하고, FC를 종속변수로 하는 단순회귀분석을 수행하여 아래과 같은 결과를 얻었다. 각 독립변수에 대한 유의성 검증에서는 '출력 데이터 유형의 수' 가 가장 유의한 변수로 나타났고(Prob>|T|: 0.0001), 다른 변수들도 충분히 유의한 것으로 나타났다.

FC = 1952 + 6.81 * 입력데이터 유형의수

(R^2 = 0.2599, Prob>|T|: 0.0437)

FC = 982 + 9.44 * Entity type수

(R^2 = 0.5091, Prob>|T|: 0.0016)

FC = 955 + 22.28 * 알고리즘수

(R^2 = 0.3815, Prob>|T|: 0.0003)

FC = 1149 + 6.79 * 출력데이터 유형의 수

(R^2 = 0.9132, Prob>|T|: 0.0001)

분석의 결과 정도의 차이는 있으나, 위의 4 가지 기능 유형 모두 의미있는 FC의 설명 변수가 됨을 알 수 있다. 따라서, 도출된 회귀식을 이용하여 기능수를 추정하는 모형을 적극적으로 검토할 것을 제안한다. 이 모형은 앞서 논의한 모형 II와 달리 알고리즘의 수를 고려하고, 입력 및 출력 유형의 수를 이용하는 일반화된 모형으로서 범용 소프트웨어 개발환경에 잘 적용될 수 있을

것이다.

변화된 소프트웨어 개발환경을 반영하고, 복잡도에 대한 각 영향도 요인의 상대적인 난이도를 반영하여 도출한 17개의 복잡도 요인을 이용하여 보정한 기능점수(FP)를 독립변수로 사용하여 분석한 결과는 다음과 같다.

선형모형과 지수모형 각각에 대하여 분석한 결과 선형모형이 보다 설명력이 높은 것으로 나타났다.

소요공수 (표준 MM) = 434 + 0.191 * FP

(R^2 = 0.5104, Prob>F: 0.0001)

소요공수 (표준 MM) = 8.36 $(FP)^{0.555}$

(R^2 = 0.3489, Prob>F: 0.0024)

이 결과는 복잡도 요인 12 개를 사용한 기능점수를 이용하여 분석한 모형 II 의 결과와 거의 차이가 없는 것으로서, 복잡도 요인을 사용하는 것에 대해 다음과 같은 사항을 고려하게 한다.

첫째, 복잡도 요인을 많이 사용하는 것의 효과성에 대해 보다 정밀한 검증이 요구된다. 요인을 많이 사용하면 예측의 정확성이 높아지게 된다. 그러나 요인 측정에 많은 시간을 소요하게 되며, 모형을 복잡하게 만들기 때문에 신중하게 추가 요인들을 모형에 편입해야 할 것이다.

둘째, 복잡도 요인 전체에 대해 측정의 정확도를 유지하기 위한 상세 기준이 필요하다. 엄격한 판단기준을 제시하고, 이 기준을 준수하는 측정치를 수집하여 추가로 정밀한 분석이 요구된다.

4.3 개발기간 및 소요비용에 대한 분석

먼저 소요공수를 이용하여 개발기간을 추정하는 관계식을 분석하였다. 선형모형과 지수모형 모두에 대하여 분석을 수행한 결과 아래와 같이 두 모형의 설명력이 비슷하게 나타났다.

개발기간(월) = 8.3 + 0.0034 * MM

(R^2 = 0.4780, Prob>F: 0.0001)

개발기간(월) = 1.36 $(MM)^{0.324}$

(R^2 = 0.4868, Prob>F: 0.0001)

이번에 수집된 데이터로는 설명력이 낮고 모형간의 차이가 분명히 드러나지 않는다. 그러나 COCOMO등 대다수 연구들이 지수모형을 채택하고 있으며, 또한 지수모형의 승수 계수도 대체로 0.35 내외로 나타나고 있어, 본 연구의 승수계수도 이에 근접하는 연구 결과로 판단된다. 따라서 개발기간에 대한 승수 모형을 우리나라의 일반 모형으로서 적극적으로 검토해볼 수 있을 것이다.

마지막으로, 실무자들 사이에서 아직도 가장 많이 사용되는 스텝수에 의한 소프트웨어 규모와 기능점수와의 관계식을 도출하였다. 도출된 관계식은 다음과 같다.

스텝수 (단위: 1000스텝) = - 72 + 0.089 * FP (R^2 = 0.8949, Prob>F: 0.0001)

이 결과는 3 세대 언어를 기준으로한 기존의 연구 결과와 매우 유사하며[18], 개선된 기능점수 방식으로 완전히 전환하기 전에 과도기적으로 스텝수를 병행하여 사용할 때 유용하게 활용할 수 있는 결과이다.

5. 결 론

본 연구는 다음과 같은 측면에서 기능점수의 예측능력과 실용성 증대에 기여하였다.

우선, 변화하는 소프트웨어 개발환경에 맞는 바람직한 기능점수 모형을 개발하고, 이를 국내의 프로젝트 실데이터를 이용하여 최초로 검증하였다. 모형 II와 모형 IV는 이러한 목적으로 개발된 새로운 기능점수 모형으로서 36개의 프로젝트 실제 데이터를 이용하여 타당성이 입증되었다. 향후 많은 활용을 거쳐 표준화된 기능점수 모형으로 발전이 기대된다.

또한 본 연구는 국내 최초로 실제 프로젝트 데이터를 이용하여 국제적으로 사용되고 있는 기능점수 모형의 활용성을 평가하였다. 국내에서 최근에 수행된 프로젝트 데이터를 이용하여 분석한 결과 IFPUG중심의 기능점수 모형은 예측의 정확성을 높이기 위해 기능 유형 및 기능수 측정에 대한 엄격하고 정밀한 기준 및 가이드가 필요함을 보여주고 있다. 앞서 연구 결과에서 언급되었듯이 기능점수 방법은 한 조직내에서 사용할 경우는 적은 오차를 보이며 신뢰성있는 기준으로 이용될 수 있지만, 서로 다른 조직간에 이용될 때는 많은 오차를 보이는 것으로 나타났다. 각 조직에서 기능점수 모형을 활용할 때 사용할 수 있도록 정의된 각 기능 유형에 대하여 우리나라의 소프트웨어 개발 관행에 맞는 용어로 측정 가이드를 작성하여 이를 지키도록 유도해야 할 것이다.

향후의 연구과제로는 클라이언트/서버 시스템의 견적 정밀도를 향상시키고 객체 지향 개발 기법 사용, 패키지 토착화(customization) 등 신 기술과 개발환경을 포괄하는 일반적인 구조로 기능점수 모형을 발전시키는 노력이 필요하다. 우선 클라이언트/서버 시스템의 개발 비용예측 정밀도를 향상하기 위해서는 본 연구에서 제시된 모형을 사용하면서 사례축적을 충실히 하는 것이 필수적이다. 또한 도구나 플랫폼 등의 발전이 빨라, 견적 방법을 안정시키기 어려운 측면을 해결하기 위하여 보다 추상화된 모형의 개발이 요구된다. 객체지향 기법의 사용을 포함하여 모든 소프트웨어 개발 프로젝트의 견적 정밀도를 높이기 위해서는 소프트웨어 개발(생산) 체계의 표준적인 확립이 필요하다. 개발 방법론의 확립은 물론이고 프로젝트관리, 도구 및 기법의 사용에 대한 체계적인 연구가 병행되어야 할 것이다.

참 고 문 헌

[1] 김영태, 이주헌 외 10인, 소프트웨어 개발비 산정기준 문제점 분석 및 개선방안에 관한 연구, 한국과학기술연구원 정책기획본부, 1993.4

[2] 김현수, 소프트웨어 개발비용 추정안내서, (주)데이콤, 1987.12

[3] 西山茂, 정보처리학회 역, 소프트웨어 규모 견적 기술의 동향, 정보처리, 제1권 제3호 (1994), pp.95-104

[4] 유병배, 김현수 외 18인, 소프트웨어 개발 및 구축기술 대가기준에 관한 연구, 한국소프트웨어산업협회, 1996.5

[5] 이주헌, "시스템통합 사업대가 산정기준에 관한 연구", 전문가협의회 회의자료, 1996.4.

[6] 田中 淳, "클라이언트/서버의 어려운 문제 - 개발공수 견적에 도전 -", 일경컴퓨터, 1995.3.20, pp.114-125.

[7] Albrecht, Allan J., "Measuring Application Development Productivity," in *Proc. IBM Application Develop. Symp.*, Monterey, CA, Oct 14-17, 1979, GUIDE Int. and SHARE, Inc., IBM Corp., pp.83-92

[8] Albrecht, Allan J., and John Gaffney, "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation", *IEEE Transactions on Software Engineering*, Vol.9, No.6(1983), pp.639-648

[9] Arthur, L.J., *Measuring Programmer Productivity and Software Quality*, John Wiley & Sons, New York, 1985.

[10] Bailey, J.W. and V.R. Basili, "A Meta-Model for Software Development

and Resource Expenditures." *Proceedings of the 5th International Conference on Software Engineering. New York: Institute of Electrical and Electronics Engineers*, 1981, pp. 107-116.

[11] Boehm, B.W., *Software Engineering Economics*, Prentice-Hall, 1981.

[12] DeMarco, T., *Controlling Software Project Management, Measurement and Estimation*, Yourdon Press, 1982.

[13] Jones, C., A Short History of Function Points and Feature Points, Software Productivity Research Inc., Burlington, MA, 1986.

[14] Kemerer, Chris F., and Benjamin S. Porter, "Improving the Reliability of Function Point Measurement: An Empirical Study", *IEEE Transactions on Software Engineering*, Vol.18, No.11(1992), pp.1011-1024

[15] Low, Graham C. and Ross Jeffery, "Function Points in the Estimation and Evaluation of the Software Process", *IEEE Transactions on Software Engineering*, Vol.16, No.1(1990), pp.64-71

[16] Mendelson, Haim, *The Economics of Information System Management*, A Draft of a Book, 1988.

[17] Mukhopadhyay, Tridas, and Sunder Kekre, "Software Effort Models for Early Estimation of Process Control Applications", *IEEE Transactions on Software Engineering*, Vol.18, No.10(1992), pp.915-924

[18] Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, McGraw Hill, 3rd ed., 1992

[19] Putnam, Lawrence H., "A General Empirical Solution to the Macro Software Sizing and Estimation Problem", *IEEE Transactions on Software Engineering*, Vol.4, No.4(1978), pp.345-361

[20] Symons, Charles R., "Function Point Analysis: Difficulties and Improvements", *IEEE Transactions on Software Engineering*, Vol.14, No.1(1988), pp.2-11