IE Interfaces
Vol. 17, Special Edition, pp. 122-130, December 2004.

생산성 기반의 소프트웨어 유지보수 대가산정 모델

배 준 수†

전북대학교 산업정보시스템공학과

A Model for Software Maintenance Cost Estimation based on Productivity

Joonsoo Bae

Department of Industrial and Information Systems Engineering, Chonbuk National University, Jeonju, 561-756

Since the cost of software maintenance occupies about 50~75% in a general successful organization, the software maintenance plays an important role in software life cycle. In particular, if the managed system needs to be operated in a long term or the system is very large and complex, then the maintenance is especially more important. Software maintenance is defined as software modification activities after customer delivery, such as improvement of performance or functionality, error correction, adaptation to environmental changes, etc. In this paper, software cost estimation models are proposed, that is based on productivity of manpower in maintenance projects. In order to do this, the activities of maintenance are classified into function change, non-function change, user support and application operation. The proposed models are constructed and verified based on the real size and cost information of projects in the real world. The approach in this paper is to discriminate the heterogeneous activities in maintenance projects, and then to calculate the respective cost of each discriminated activity. By using the proposed models, the total cost of maintenance project is summed from the costs of four activities. In addition the number of conflicts between owner and order receiver about the amount of cost will be reduced and the reasonable cost estimation system will be established.

Keywords: software maintenance, cost estimation model, productivity

1. 서 론

소프트웨어 유지보수란 소프트웨어를 고객에게 인도한 후에 그 소프트웨어를 변경시키는 활동을 말한다. 그 활동은 주로 프로그램의 소스 코드를 변경하는 것이며, 그 목적은 오류를 수정하거나, 성능이나 기능을 개선하거나, 또는 변화된 환경에 제품을 적응시키기 위한 것이다(Longstreet, 1990; Martine, 1983; Vliet, 1993). 성공적인 소프트웨어의 경우 유지보수 비용은 소프트웨어 개발비의 50~75%를 차지할 정도로(Sommerville, 1996; Vliet, 1993), 유지보수는 소프트웨어의 성공적인 도입 및 활용에 중요한 활동이다.

소프트웨어 공학에서 유지보수 활동은 소프트웨어 수명주기 프로세스의 한 가지 과정으로 간주된다. 소프트웨어의 개발 및 운영에 관한 가장 일반적인 프레임워크를 다루는 표준은 ISO(International Organization for Standardization)와 IEC(International Electrotechnical Commission)가 공동으로 1995년에 제정한

이 논문은 2004년도 한국전산원의 지원에 의하여 연구되었음.
†연락저자 : 배준수 교수, 561-756 전북 전주시 덕진구 덕진동 전북대학교 산업정보시스템공학과, Fax : 063-270-2333
E-mail : jsbae@chonbuk.ac.kr

"ISO/IEC 12207-소프트웨어 수명주기 프로세스"인데, 여기서는 유지보수 프로세스를 <그림 1>과 같이 여섯 가지 활동으로 구성하고 있다(ISO/IEC, 1995). 미국전기전자학회(IEEE)의 "IEEE 1219-소프트웨어 유지보수에 관한 IEEE 표준"은 유지보수 프로세스를 7단계로 구분하고, 각 단계에 필요한 입력값/프로세스/통제/출력값을 설명하고 있다. 이러한 소프트웨어 유지보수의 프로세스도 일반적인 시스템 개발단계와 마찬가지로 "문제의 정의 및 계획 - 분석 및 설계 - 구현 - 시험 - 인도"로 정리할 수 있다(Martine, 1983; NCA, 2002).

본 논문에서는 소프트웨어 유지보수에서 수행하는 활동들의 유형에 관하여 알아보고, 이러한 유지보수의 유형별 적절한 대가산정 모형을 통하여 전체 유지보수 대가를 산정하는 방법에 대하여 살펴본다.

2. 소프트웨어 유지보수의 유형

ISO/IEC의 소프트웨어 유지보수 활동에 관한 표준인 "ISO/IEC 14764-소프트웨어 유지보수"는 소프트웨어 유지보수 프로세스에 관한 상세한 과정과 함께 유지보수 유형을 제시한다(ISO/IEC, 1999). 이 표준에서 유지보수 대상 소프트웨어에 대해 요구되는 변경내역을 "변경요청(MR: Maintenance Request)"이라고 부르는데, 변경요청은 소프트웨어의 결점을 보완하는 수정(correction)과, 신규 또는 변경된 사용자 환경이나 요구를 반영하는 개선(enhancement)으로 구분되고, 특히 개선은 비용이나 시간적으로 많이 소모될 수 있으며 유지보수 활동에서 많은 비중을 차지하는 활동이다(Chapin, 2001).

IEEE 1219에서도 소프트웨어 유지보수를 4가지로 분류하고 있는데, 오류수정, 환경적응, 완전개선은 동일하나, 예방조치 대신에 긴급조치(emergency maintenance) 유형을 정의하고 있다. 긴급조치란 시스템이 계속 작동하기 위하여 예상치 못한 오류를 수정하는 활동을 의미한다(IEEE, 1998).

국제 기능점수 사용자 그룹인 IFPUG(International Function Point User Group)의 CPM(Counting Practice Manual) 모델은 NESMA 모델(Martine, 1983)과 함께 개선(enhancement)을 위한 기능점수 규모를 산정하는 대표적인 방법을 제공하고 있다.

이 외에도 국제 소프트웨어 벤치마킹 사용자 그룹인 ISBUG (International Software Benchmarking User Group)에서는 소프트웨어 개발 및 관리 모델을 크게 다섯 가지로 분류하고 있다(ISBSG, 2002). 영국의 소프트웨어 기준 협회인 UKSMA(United Kingdom Software Maintenance Association)에서도 소프트웨어 유지보수와 관련된 기반 모델을 ISBSG와 마찬가지로 유지보수

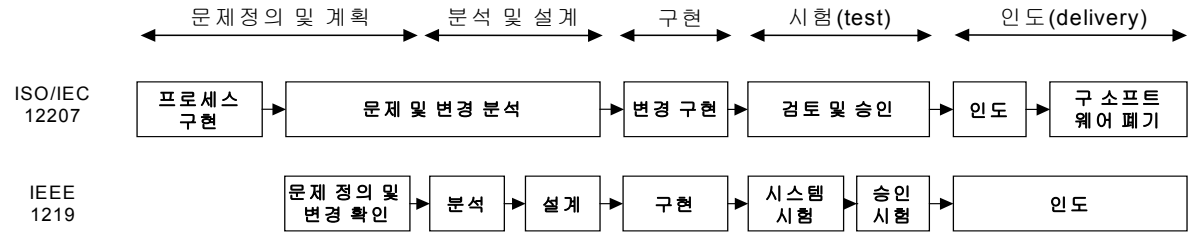

그림 1. 소프트웨어 유지보수를 위한 표준 프로세스.

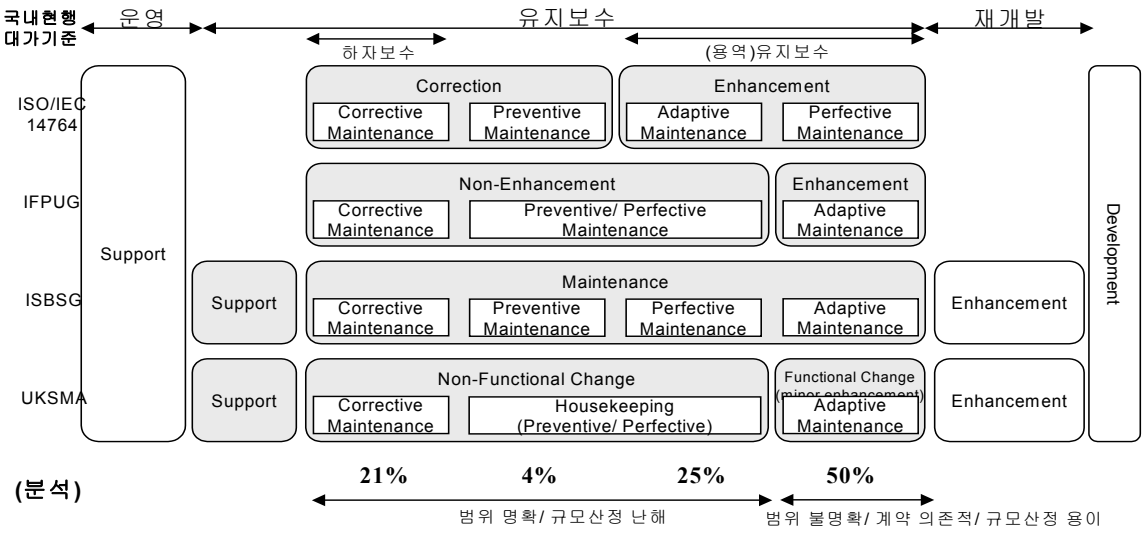

그림 2. 표준화 기구들의 소프트웨어 유지보수 유형 비교.

(maintenance)와 지원(support)을 구분하고 있으며, 특히 유지보수를 기능변경이 발생하는 기능개선(adaptive maintenance) 유형과, 기능변경이 발생하지 않는 운용(perfective/preventive maintenance) 및 오류수정(corrective maintenance) 유형으로 나누고 있다(UKSMA, 2001).

위에서 언급한 대표적인 표준화 기관에서 정의한 유지보수의 정의와 유형을 정리해 보면 <그림 2>와 같다. 이렇게 각 표준을 분석해 본 결과 소프트웨어 개발 이후에 이루어지는 개선 및 서비스(evolution and service) 영역은 크게 운영, 유지보수, 사용자 지원, 재개발로 나눌 수 있고, 그 중에서 유지보수의 유형은 크게는 기능변경과 비기능변경으로 나눌 수 있다는 것을 알 수 있다.

3. 소프트웨어 유지보수 대가산정 모형

3.1 국내의 소프트웨어 유지보수 대가산정 모형

국내의 소프트웨어 유지보수 대가산정에 관한 기준은 정보통신부 고시로 규정된 "소프트웨어 사업대가 기준"에 포함되어 있는데, 최근본은 2004년 2월에 개정 고시되었다(MIC, 2004). 연간 소프트웨어 용역 유지보수의 대가는 소프트웨어 개발비 산정가의 10%~15% 범위 내에서 결정하는데, 이 범위를 결정하는 다섯 가지 요소는 (i) 유지보수 횟수(연간 유지보수를 실시하는 횟수의 정도), (ii) 자료처리 건수(처리하는 자료의 건수의 정도), (iii) 타 시스템 연계(몇 개의 다른 시스템과의 연계가 필요한지 정도), (iv) 실무지식 필요(업무에 관한 기초지식이나 전문실무 능력이 필요한지의 정도), (v) 분산처리 여부(통합하의 분산처리인지 순수 분산처리인지)로 구성된다. 그러나 이 기준에서는 유지보수 활동유형에 대해 명확한 정의가 없고, 각 유형별 대가모형을 제시하지 못하고 있다. 또한 유지보수의 활동규모가 대가에 반영이 되지 못한다는 문제점과 상한과 하한이 고정되어 있다는 문제점이 있다.

3.2 COCOMO II

COCOMO(COnstructive COst MOdel) 모델은 소프트웨어 개발에 소요되는 공수를 예측하고 비용을 산정하기 위하여 Boehm에 의해 1981년에 제안되었으며, 발전된 소프트웨어 기술을 반영하여 수정된 COCOMO II 모델이 1995년에 발표되었다(Boehm, 2000). 이 모델에서 소프트웨어 유지보수는 기능의 추가 및 확장을 포함하며, 기존 소프트웨어의 50% 이상 변경이나 독립적인 인터페이스 개발은 제외하고 있다.

COCOMO 모델에서 유지보수의 규모 $Size_M$ 는 다음 식과 같이 LOC, 기능점수, 객체점수와 같은 기본 코드규모(BaseCode-Size)에 연간 유지보수 비율 MCF(Maintenance Change Factor)와 보정계수 MAF(Maintenance Adjustment Factor)를 곱하여 계산된다.

$$(Size_M)=[(\text{기본코드규모})\times MCF]\times MAF$$

단, $MCF=\frac{(\text{추가규모})+(\text{변경규모})}{(\text{기본코드규모})}$,

$$MAF=1+\frac{(\text{소프트웨어 이해도})}{100}\times(\text{프로그래머 비친숙도})$$

이를 바탕으로 유지보수 비용은 아래와 같이 계산된다.

$$PM_M=A\times(Size_M)^E\times\prod_{i=1}^{15}EM_i$$

$$E=B+0.01\times\sum_{j=1}^{5}SF_j$$

A는 비용계수로 COCOMO II에서는 2.94이며, E는 규모지수로 유지보수 규모에 적용되며, 비용조정계수인 EMi는 생산성 인자(신뢰성, DB 규모, 복잡성, 문서화 정도), 플랫폼 인자(실행기간, 저장공간 제한, 플랫폼 안정성), 인적 요인(분석자 능력, 프로그래머 능력, 구성원 지속성, 애플리케이션 경험, 플랫폼 경험, 언어/툴 경험), 제품요인(사용 툴, 다중 사이트, 개발일정)에 의해 결정된다. B는 0.91이며 SF는 Scale Factor를 나타낸다.

3.3 NESMA

네덜란드 소프트웨어 측정 위원회인 NESMA (Netherlands Software Metrics Users Association)는 IFPUG CPM을 발전시켜 소프트웨어 개선을 위한 기능점수 분석 모델을 제안하였다(NESMA, 2001). 소프트웨어 개선 프로젝트의 기능점수 분석은 추가, 삭제, 변경되는 데이터 및 트랜잭션 기능을 측정하여 유지보수 규모를 미조정 기능점수로 계산하고, 조정인자를 적용하여 조정된 기능점수를 산출한다. NESMA의 유지보수 기능점수 계산에서 가장 차별적인 요소는 삭제되는 기능에 대하여 가중치를 0.4를 주고 있다는 점이다.

미조정 기능점수(UEFP; Unadjusted Enhanced Function Point)는 조정인자 VAF(Value Adjusted Factor)를 통하여 유지보수 프로젝트의 기능점수(EFP; Enhanced Function Point) 규모로 조정되는데, 이 조정인자는 14가지의 시스템 특성을 통하여 평가된다.

유지보수 이후의 조정인자 (VAF_{AFTER})는 추가, 변경된 기능에 적용되고, 유지보수 이전의 조정인자 (VAF_{BEFORE})는 삭제된 기능에 적용된다.

$$EFP_{project}=\sum UEFP_{ADDED}\times VAF_{AFTER}+\sum UEFP_{CHANGED}\times VAF_{AFTER}+\sum UEFP_{DELETED}\times VAF_{BEFORE}$$

3.4 IFPUG

IFPUG(International Function Point Users Group)에서는 소프트웨어를 기능변경하는 경우에 사용할 수 있는 기능점수 산정방

법을 제공하고 있지만, 대가로의 변형에 대해서는 별도 언급이 없다(IFPUG, 2004). 또한, 기능변경 시의 일반 시스템 특성은 개발 시와 동일하게 사용하며, 기능변경 전후의 시스템 특성이 변한다고 가정하고 있다.

기능변경 프로젝트의 기능점수 계산에는 프로젝트에 대한 사용자 요구사항이 반영된 애플리케이션 기능(추가, 변경, 삭제된 기능점수), 프로젝트에 대한 사용자 요구로 변환된 기능점수, 애플리케이션 조정인자(VAF)를 통하여 구성된다.

개선 프로젝트의 기능점수(EFP)는 다음과 같이 계산된다. ADD, CHGA, CFP, DEL은 각각 추가, 변경, 변환, 삭제된 기능점수이며, VAFA, VAFB는 개선 전후의 조정인자이다.

$$EFP = (ADD + CHGA + CFP) \times VAFA + DEL \times VAFB$$

조정인자(VAF)는 대상 애플리케이션에 대하여 14가지 시스템 특성을 기준으로 결정된다. 그 시스템의 특성으로는 데이터 통신, 분산데이터 처리, 시스템 성능, 자원 제약 정도, 트랜잭션 비율, 온라인 데이터 입력, 최종 사용자 효율성, 온라인 갱신, 처리 복잡도, 재사용성, 설치 용이성, 운영 용이성, 다중 설치성이 있으며, 각각 6등급으로 판정된다.

3.5 비용산정 패키지

소프트웨어 프로젝트의 계획과 평가를 위해 PRICE 사에서 개발된 PRICE-S(PRICE Software Estimating Model)는 모든 소프트웨어 프로젝트에 적용 가능하며 시스템 개념과 운영 테스트를 포함한 전 수명주기를 고려하고 있다.

PRICE-S는 프로그램 개발 대가 LH(Labor Hours)를 산정하기 위하여 소프트웨어의 볼륨(VOL)과, 소프트웨어 프로그램과 실제 운영의 생산성, 효율성에 관련된 보정계수 PROFAC(Productivity Factor)를 사용한다.

$$LH = \frac{[e^{PROFAC}][VOL^{f(PROFAC)}]}{1000}$$

PRICE-S는 설계, 코드, 테스트의 소프트웨어 개발 3단계에 해당되는 대가를 산정하기 위하여 다섯 가지 활동(시스템 엔지니어링, 프로그래밍, 외형, 품질조회 문서와 프로그램 운영)으로 세분화한다.

PRICE-S는 수명주기 모델에서 소프트웨어 유지보수 규모를 산정하기 위하여 지원 스케줄, 설비의 개수, 기능개선 수준, 소프트웨어 품질 수준, 유지보수 생산성 보정요소를 사용한다.

한편, Galorath 사에서 제공되는 도구인 SEER-SEM은 모든 소프트웨어 수명주기에 적용될 수 있다. SEER-SEM은 규모, 지식기반 입력값, 입력인자를 기본 입력값으로 사용한다. 규모는 다른 모델과 마찬가지로 LOC, 기능점수, 객체점수를 사용할 수 있으며, 지식기반 입력값은 해당 모델에서 사용할 지식기반을 명세하는데, 예를 들면, 플랫폼, 애플리케이션, 소프트웨어 획득 방법(개발, 수정, 재개발), 소프트웨어 개발방법(폭포수 모형, 진화적, 나선형), 개발표준 등을 명세한다. 또한, 개인적 능력과 경험, 개발지원 환경, 제품개발 요구, 재사용성 요구, 개발환경 복잡도, 기술과 환경인자의 민감도 등 30가지 이상의 입력인자도 반영한다.

SEER-SEM은 운영, 시스템 엔지니어링, 설계, 코드, 데이터, 테스트, 형상관리, 품질보증의 다양한 대가산출 모델을 제공한다. 또한, SEER-SEM은 연간 비용과 공수를 제공하는 유지보수 모델을 포함하며, 사용자가 지원주기를 연간 변화비율, 지원사이트 수, 기대 프로그램 성장 등 다른 지원인자와 같이 명세할 수 있도록 한다.

4. 유지보수 대가산정 모형의 고찰

먼저 유지보수 대가를 산정하는 가장 기본적인 개념은 ACT (Annual Change Traffic)에서 출발한다. ACT 모델은 Boehm이 제안하여 COCOMO 81 모델에서 적용된 유지보수 대가 모델로 아래와 같이 소프트웨어 연간 변화량(ACT)에 의해서 개발 투입공수에 비례하여 유지보수 투입공수를 산정하고자 한다.

*(연간 유지보수 투입공수) = 1.0*ACT*(SW 개발 시 투입공수)*

또는, 투입공수에 비례하여 대가가 결정된다고 가정하면 아래와 같이 확장할 수 있다.

*(연간 유지보수 비용) = 1.0*ACT*(SW 개발비용)*

이 모델은 유지보수 활동의 비용요인이 개발활동의 비용요인과 동일하다고 가정하고 있으므로, 전체 소프트웨어의 규모 중 변화비율인 ACT에 따라서 개발비에 비례하여 유지보수 대가가 발생할 것이라고 설명한다. 그러나 이 모델의 가장 큰 문제는 유지보수와 개발의 비용요인 차이로 인하여, 유지보수 대가가 개발비에 비례한다고 가정하기가 힘들다는 것이다.

두 번째 방식은 위의 문제를 개선하기 위한 방법으로 COCOMO II처럼 소프트웨어 유지보수 규모로부터 직접 유지보수 대가를 산정하는 방법이다.

(연간 유지보수 투입공수) = A(보정된 유지보수 규모)^B*

이처럼 소프트웨어 규모에 의하여 대가를 추정하는 경우에는 멱함수(Power Function)를 사용하여 규모의 경제를 반영하는 모델이 많이 쓰이며, COCOMO II 모델 외에도 기능점수 규모 기반의 IFPUG 모델이나 NESMA 모델에서도 규모에 기반한 개발비 산정 및 유지보수 산정에서 적용되고 있다.

본 연구에서는 유지보수 대가산정 모델을 도출하기 위하여 위의 두 가지 접근 방식을 모두 고려한다. 특히, 전체 유지보수 사업을 구성하는 활동의 유형에 따라서 어떤 형태가 더 적합한지에 대해 검토하는 방식으로 모델을 수립하고자 한다.

5. 유지보수 대가산정의 회귀모형

본 연구에서는 유지보수 업무를 외주로 계약하는 40건의 유지보수 비용자료를 수집하였으며, 앞에서 제시한 유지보수 사업의 대가산정 방식을 반영하여 회귀모형을 도출하기 위하여 <표 1>과 같이 네 가지 실험을 하였다. 독립변수로는 대상 시스템의 초기 개발비와, 기능점수로 환산된 시스템의 규모 두 가지를 사용하였으며, 종속변수로 유지보수 활동의 각 유형에 대한 계약금액과 투입공수를 적용하였다.

표 1. 회귀모형의 실험계획

실험	독립변수	종속변수
실험 1	대상 시스템의 초기 개발비(만원)	유지보수 사업의 유형별 계약금액(만원)
실험 2		유지보수 사업의 유형별 투입공수(MM)
실험 3	대상 시스템의 규모(기능점수)	유지보수 사업의 유형별 계약금액(만원)
실험 4		유지보수 사업의 유형별 투입공수(MM)

유지보수 사업의 유형별 계약금액은 총 계약금액, 순수 유지보수비, 사용자지원비, 애플리케이션 운영비, 기능변경 유지보수비, 비기능변경 유지보수비를 말하며, 유형별 투입공수도 대응되는 6가지 투입공수를 말한다. 즉, 각 실험당 총 6가지의 독립변수, 종속변수 쌍을 대상으로 회귀분석을 시행하며, 회귀함수는 앞 절에서 언급한 선형함수와 멱함수(Power function)에 대하여 분석하였다.

<표 2>에 위의 네 가지 실험에 대한 선형함수와 멱함수에 대한 회귀모형의 적합도에 대하여 정리를 하였다. 실험 1이나 실험 2의 결과를 보면, 현재 수집된 데이터 상으로는 초기 개발비를 기준으로 유지보수 사업의 계약금액이나 실제 투입공수를 추정하는 것은 거의 힘들다고 할 수 있다. 반면에 대상 시스템의 규모인 기능점수를 통하여 유지보수 사업을 예측하는 것, 특히 투입될 공수에 대하여 예측하는 것은 실험 4와 같이 의미가 있음을 알 수 있었다.

실험 1과 실험 2의 결과를 살펴보면, 유지보수 대상 시스템의 개발비만을 사용하여 유지보수 사업의 대가를 산정하는 것이 힘들다는 것을 보여준다. 유지보수 대상 소프트웨어의 개발비를 유지보수 사업의 대가를 추정하는 데 사용하기 힘든 이유는 다음과 같이 예상된다. 가장 근본적인 원인은 소프트웨어 개발과 소프트웨어 유지보수 사업의 비용요인이 차이가 크기 때문으로 예상된다. 시스템 개발비에 근거하기 보다는 개발된 시스템의 규모나 특성에 의하여 유지보수에 필요한 금액(실험 1)이나 인력(실험 2)이 산정되고 계약되었음을 나타내고 있다. 그러나 데이터의 수가 적고, 수집된 데이터의 비용분할 과정에서 왜곡이 되었을 가능성에 대해서도 배제할 수는 없다.

반면에 유지보수 대상 시스템의 규모를 독립변수로 사용한 실험 3과 실험 4의 경우를 살펴보면, 상대적으로 유지보수 사업의 계약금액과 투입공수를 더 잘 반영하고 있다. 이는 소프트웨어 개발대가를 산정하기 위하여 보정되기 이전의 시스템의 규모가 소프트웨어 유지보수 계약금액이나 투입공수를 결정하는 데 더 큰 관련성이 있음을 보여준다. 즉, 소프트웨어 개발비는 대상 시스템의 특성과 계약 당사자들의 여건에 의하여 왜곡되어 있는 데이터임에 비하여, 보정 전인 소프트웨어의 규모는 실제로 소프트웨어 유지보수의 대가나 투입공수의 크기를 결정하는 데 영향을 크게 미쳤음을 추측할 수 있다.

실험 4의 결과를 좀더 자세히 살펴보자. 결정계수가 60% 이상인 경우와, F-통계량이 유의수준 5% 이내에서 유의한 모형의 경우 *로 표시되어 있다. 실험 4는 보정 전인 시스템의 규모

표 2. 회귀모형의 적합도 비교. (* 표시는 결정계수가 60% 이상, F-통계량은 유의수준 5% 이내에서 유의한 경우)

활동-유형 \ 실험		실험 1			실험 2			실험 3			실험 4		
		R^2	F	Sig.	R^2	F	Sig.	R^2	F	Sig.	R^2	F	Sig.
총 사업	Linear	0.087	0.47	0.522	0.127	0.73	0.433	0.338	3.06	0.131	*0.807	25.08	*0.002
	Power	0.064	0.34	0.585	0.214	1.36	0.296	0.264	2.16	0.192	0.410	4.17	0.087
유지보수	Linear	0.105	0.58	0.479	0.149	0.87	0.393	0.066	0.42	0.539	*0.764	19.47	*0.005
	Power	0.082	0.45	0.534	0.246	1.63	0.258	0.115	0.78	0.412	0.344	3.15	0.126
기능변경	Linear	0.139	0.81	0.410	0.174	1.05	0.352	0.139	0.97	0.363	*0.826	28.44	*0.002
	Power	0.160	0.95	0.373	0.224	1.44	0.284	0.070	0.45	0.526	0.440	4.72	0.073
비기능변경	Linear	0.008	0.04	0.846	0.042	0.22	0.661	0.579	8.26	0.028	0.594	8.79	0.025
	Power	0.041	0.22	0.662	0.024	0.12	0.742	0.438	4.68	0.074	*0.766	19.61	*0.004
사용자지원	Linear	0.239	1.57	0.265	0.286	2.00	0.217	0.109	0.73	0.425	0.498	5.96	*0.050
	Power	0.409	3.46	0.122	0.549	6.08	0.057	0.027	0.17	0.698	0.174	1.26	0.304
App.운영	Linear	0.301	2.15	0.202	0.324	2.40	0.182	0.466	5.23	0.062	*0.664	11.85	*0.014
	Power	0.480	4.62	0.084	0.549	6.08	0.057	0.182	1.34	0.292	*0.773	20.42	*0.004

인 기능점수를 독립변수로 사용하고, 유지보수 활동의 투입공수를 종속변수로 사용하였다. 즉, 개발 정보와 유지보수 정보 모두 계약금액의 경우 실제로 함수 관계를 추정하기 힘들지만, 계약금액으로 보정되기 이전의 대상 시스템의 규모와 유지보수의 투입공수는 함수 관계를 회귀모형으로 추정할 수 있다는 것이다.

특히, * 표시가 나타내는 것처럼 총 사업 투입공수, 유지보수 투입공수, 기능변경 활동의 투입공수는 대상 시스템의 규모를 사용하여 선형회귀모형(linear function)으로 추정할 수 있으며, 애플리케이션 운영의 투입공수와 비기능변경 활동의 투입공수는 멱함수(power function) 형태의 회귀모형으로 추정하는 것이 적합함을 보여준다. 나아가 여기서 의미하는 바는 유지보수 사업을 구성하는 네 가지 유형들(기능변경, 비기능변경, 사용자지원, 애플리케이션 운영)은 각각 비용요인이 다르기 때문에 동일한 형태의 대가모형을 사용하는 것보다, 차별화된 대가모형을 사용하여 개별적으로 대가를 추정하는 것이 바람직하다는 것이다.

유지보수 활동유형 중에서 멱함수로 회귀 가능한 비기능변경 유지보수 활동과 애플리케이션 운영은, 규모의 경제에 의하여 시스템의 규모에 대하여 체감하는 형태를 나타낼 수 있다는 것을 의미한다. 즉, 시스템의 규모가 일정 정도 이상 커지더라도 기능변경이 아닌 소프트웨어 변경이나, 애플리케이션 운영에 있어서 요구되는 투입공수는 비례하여 커지지 않는다는 것이다. 반면에 기능변경에 해당되는 유지보수 활동이나 사용자 지원 활동은 시스템의 규모가 커지면 커질수록 투입공수가 체감하지 않고 비례하여 증가한다고 해석할 수 있다.

실험 4의 결과로 도출된, 대상 시스템의 규모(기능점수, FP_{Dev})를 통하여 유지보수 활동의 투입공수를 예측하기 위한 회귀모형은 아래와 같다.

(1) 총 사업 투입공수

$MM_{total}=35.0386+0.0051*FP_{Dev}$, $R^2=0.807$

$MM_{total}=0.4627(FP_{Dev})^{0.5711}$, $R^2=0.410$

(2) 순수 유지보수 투입공수

$MM_M=24.3079+0.0031 * FP_{Dev}$, $R^2=0.746$

$MM_M=0.4079(FP_{Dev})^{0.534}$, $R^2=0.344$

(3) 기능변경 유지보수 투입공수

$MM_{M1}=-6.9108+0.0016 * FP_{Dev}$, $R^2=0.826$

$MM_{M1}=0.0329(FP_{Dev})^{0.6653}$, $R^2=0.440$

(4) 비기능변경 유지보수 투입공수

$MM_{M2}=10.1911+0.0018 * FP_{Dev}$, $R^2=0.594$

$MM_{M2}=0.0038(FP_{Dev})^{0.9373}$, $R^2=0.766$

(5) 사용자지원 투입공수

$MM_S=15.919+0.0005 * FP_{Dev}$, $R^2=0.498$

$MM_S=1.5875(FP_{Dev})^{0.2757}$, $R^2=0.174$

(6) 애플리케이션 운영 투입공수

$MM_O=-4.2762+0.0012 * FP_{Dev}$, $R^2=0.664$

$MM_O=5.10E-05(FP_{Dev})^{1.2619}$, $R^2=0.773$

또한, 위 모델들의 정확성을 평가하기 위하여 MRE(Magnitude of Relative Error)를 측정하였으며, 대표적인 세 가지 척도 MMRE(Mean MRE), MdMRE(Median MRE), PRED(PREDiction quality)에 대한 평가결과를 <표 3>에 정리하였다.

수집된 실험 데이터 상에서는 MMRE, MdMRE의 경우 기능변경과 사용자 지원은 선형모형이 더 우수한 결과를 보였으며, 비기능변경과 애플리케이션 운영은 멱함수 형태의 회귀모형이 더 우수한 결과를 보였다. 또한, PRED(25)의 경우 기능변경과 애플리케이션 운영은 선형모형이, 유지보수와 비기능변경은 멱함수모형이 더 우수하였다. 그러나 실험 데이터의 수가 확보되어야 모형의 상대적인 정확성에 대해서 유의하게 평가할 수 있을 것이다.

6. 유지보수 대가산정의 후보모형 제안

지금까지 유지보수 사업의 실제 데이터를 수집하여 유지보수 사업대가를 산정하기 위한 회귀모형들을 찾아보았다. 그러나 이러한 회귀모형은 과거의 데이터를 가장 잘 표현할 수 있는 근사한 모형일 뿐이며, 실제로 대가산정에 사용되어야 할 모형은 이와 차이가 있을 수 있다. 즉, 사용자의 추가요구에 의해 수

표 3. 대상 시스템의 기능점수와 유지보수 유형별 투입공수의 회귀 모델 정확성

		총 사업	유지보수	사용자지원	App.운영	기능변경	비기능변경
Linear Function	MMRE	0.37	0.41	0.35	0.79	0.38	0.72
	MdMRE	0.27	0.33	0.27	0.52	0.28	0.51
	PRED(25)	0.36	0.33	0.38	0.38	0.50	0.30
Power Function	MMRE	0.46	0.37	0.42	0.40	0.52	0.53
	MdMRE	0.41	0.31	0.36	0.35	0.60	0.25
	PRED(25)	0.17	0.38	0.38	0.27	0.20	0.50

행되는 기능변경의 유지보수 유형은 대상 시스템의 규모나 개발비 이외에 계약연도에 실제로 요구되는 유지보수 활동의 규모에 따라서 대가를 산정하는 것이 합리적이다. 이러한 유형을 일관된 회귀모형을 통해 기준을 제시하는 것은 계약 당사자들 간의 계약금액에 대한 유연성을 오히려 저해하여 유지보수 활동의 품질을 저해하는 요인이 된다. 또한 고정된 계약금액은, 명시되지 않은 기능변경에 대해 유지보수 사업 중에 포함시켜야 하는지의 여부에 대해 실제로 발주자와 수주자 간 마찰의 원인을 제공하고 있기 때문이다. 이러한 경우를 비롯하여, 본 장에서는 <표 4>와 같은 다섯 가지 모형을 대가산정의 후보로 채택하였다. 표에서 표기한 방식은 아래와 같다.

$Cost_{activity}$: 해당 활동유형을 수행하는 데 필요한 비용
MMactivity : 해당 활동유형을 수행하는 데 필요한 투입공수
M1 : 기능변경 유형, M2 : 비기능변경 유형
O : 애플리케이션 운영 S : 사용자지원 유형

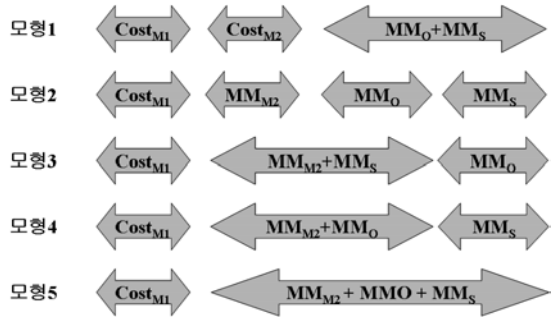

그림 3. 유지보수 대가모형의 종류.

▷ **모형** 1 - 활동규모 기반 모형

유지보수 사업의 대가를 산정하기 위하여 기능변경 유형과 비기능변경 유형으로 나누고 각각을 활동의 규모를 중심으로 대가를 산정하는 방법이다.

기능변경의 경우는 IFPUG나 NESMA에서 제안하는 것처럼 유지보수에 수행될 기능변경에 대하여 기능점수를 산정한 후, 기능점수의 단가를 곱하여 기능변경 대가를 산정하는 방법이고, 비기능변경의 경우는 시스템 개발 대비 전년도 MM 투입비율이나 코드라인 변경비율을 사용해 비기능변경 대가를 산정하는 방법이다.

▷ **모형** 2 - 생산성 기반 모형(1) : 활동 유형을 각각 분리 산정

유지보수 사업의 유형별로 대가를 세분화하고, 기능변경 유형의 대가를 제외한 비기능변경 유형, 사용자지원 유형, 애플리케이션 운영유형에 대한 공수를 각각 기능점수를 통하여 산정하는 방식이다. 이 때 사용되는 모형은 활동유형의 성격에 따라 선형모형과 멱함수모형으로 분리하여 적용한다. 여기에서도 "모형 1"과 마찬가지로 기능점수에 대한 대가는 활동을 기반으로 산정하도록 한다.

▷ **모형** 3 - 생산성 기반 모형(2) : 비기능변경과 사용자지원을 통합 산정

"모형 2"와 유사하지만, 일반적인 유지보수 활동 중 가변적인 기능변경 활동을 제외한 나머지 두 가지 활동인 비기능변경과 사용자지원 활동을 통합하여 추정하는 모형이다. 이러한 모형산정을 시도하는 이유는 기존의 고정된 유지보수 사업 내에서 두 가지 활동이 상호 교환적이거나, 계약수립 시점에서나 데이터 수집과정에서 명확하게 구분되지 않을 가능성이 있기 때문이다.

▷ **모형** 4 - 생산성 기반 모형(3) : 비기능변경과 애플리케이션 운영을 통합 산정

"모형 3"과 유사하다. 단지 비기능변경과 애플리케이션 운영을 통합 산정하는 것이 다를 뿐, 비기능변경과 애플리케이션 운영 모두 특정 규모(기능점수)의 대상 시스템에 대한 일상적인 또는 정기적, 비정기적인 소프트웨어 관리활동을 수행한다는 점에서 성격이 유사하다. 두 활동 모두 기능변경이나 사용자유지와는 달리 발주자의 요구에 의해 별 영향을 받지 않을 가능성이 있으며, 이는 후에 통계분석을 통하여 유추될 수 있다.

▷ **모형** 5 - 생산성 기반 모형(4) : 비기능변경, 사용자지원, 애플리케이션 운영을 모두 통합

앞의 세 가지 모형과 유사하다. 앞에서와 마찬가지로 활동의 상호 전환이 있었거나, 업무유형별로 명확하게 경계를 지을 수 없을 경우를 위하여, 가변적인 기능변경 유형을 제외한 나머지 활동(비기능변경, 사용자지원, 애플리케이션 운영)을 통합한 모델을 수립하는 방식이다. 이러한 통합활동은 앞에서 행한 실험에서와 같이 기능점수에 의해서 통합 투입공수를 추정하고,

표 4. 유지보수 대가산정의 후보 모형

대가산정 방식		산정 방법
활동규모 기반	모형 1	$Cost_{total} = Cost_{M1} + Cost_{M2} + (MM_O + MM_S) * (노임단가)$ $Cost_{M1} += A * (FP_{Dev})^B$, $Cost_{M2} = C * ACT * Cost_{Dev}$
생산성 기반	모형 2	$Cost_{total} = Cost_{M1} + (MM_{M2} + MM_S + MM_O) * (노임단가)$
	모형 3	$Cost_{total} = Cost_{M1} + (MM_{M2+S} + MM_O) * (노임단가)$
	모형 4	$Cost_{total} = Cost_{M1} + (MM_{M2+O} + MM_S) * (노임단가)$
	모형 5	$Cost_{total} = Cost_{M1} + MM_{M2+O+S} * (노임단가)$

투입공수당 노임단가를 적용하여 대가를 산정한다. 여기에 기능변경에 해당되는 활동규모 기반의 대가를 합산하여 전체 유지보수 사업대가를 산정한다.

위의 다섯 가지 모형 중 본 논문에서는 생산성 기반의 대가모형을 산정하는 것이므로 모형 1은 고려하지 않고, 모형 2, 3, 4, 5를 중점적으로 살펴본다. 또한 모형 2는 5장의 실험에서 회귀분석을 통하여 선형모형과 멱함수모형을 도출하였다. 여기에서는 나머지 세 가지 모형(모형 3, 4, 5)을 도출하기 위하여 개별적인 활동유형이 아니라 가능한 통합활동유형(비기능변경과 운영의 통합, 비기능변경과 사용자지원의 통합, 비기능변경과 사용자지원과 애플리케이션 운영을 통합)을 적용하는 회귀모형을 제시한다.

(1) 모형 3: (비기능변경+사용자지원) 투입공수

$MM_{M1+S}=12.8621+0.0021*FP_{Dev}$, $R^2=0.965$

$MM_{M1+S}=0.0401(FP_{Dev})^{0.7327}$, $R^2=0.823$

(2) 모형 4: (비기능변경+APP.운영) 투입공수

$MM_{M1+O}=-3.7987+0.0026*FP_{Dev}$, $R^2=0.958$

$MM_{M1+O}=0.0078(FP_{Dev})^{0.8797}$, $R^2=0.784$

(3) 모형 5: (비기능변경+사용자지원+APP.운영) 투입공수

$MM_{M1+S+O}=3.6304+0.0032*FP_{Dev}$, $R^2=0.969$

$MM_{M1+S+O}=0.0169(FP_{Dev})^{0.8383}$, $R^2=0.841$

표 5. 유지보수 대상 시스템의 규모(기능점수)와 세 가지 통합활동 투입공수의 회귀 모델 정확성

		통합활동 1	통합활동 2	통합활동 3
Linear Function	MMRE	0.23	0.30	0.23
	MdMRE	0.24	0.34	0.29
	PRED(25)	0.57	0.38	0.50
Power Function	MMRE	0.26	0.28	0.26
	MdMRE	0.27	0.24	0.24
	PRED(25)	0.43	0.63	0.50

(통합활동 1)= (비기능변경+사용자지원) 투입공수
(통합활동 2)= (비기능변경+애플리케이션 운영) 투입공수
(통합활동 3)= (비기능변경+사용자지원+애플리케이션 운영) 투입공수

위 <표 5>에서 두 번째 통합활동은 멱함수 형태의 회귀식이 우수하고, 첫 번째와 세 번째의 통합활동은 선형회귀식이 약간 우수한 것을 보여주고 있다. 이러한 결과는 새로운 모형의 도출이 의미 있음을 보여준다. 즉, 네 가지 세부적인 유지활동 유형(기능변경, 비기능변경, 사용자지원, 애플리케이션 운영) 중에서 기능변경은 가변적이므로 기능점수와 같은 유지보수 규모를 산정하여 직접 대가를 산정하고, 비기능변경과 애플리케이션 운영은 통합하여 멱급수 회귀모형을 통하여 공수를 산정하며(MMRE=0.28, PRED(25)=0.63), 사용자지원 유형은 별도의 선형모형을 통하여 공수를 산정한 후(MMRE=0.35, PRED(25)=0.38), 총 유지보수 사업의 대가산정을 하는 방법인 모델 4에 대하여 정확성을 검토하였다.

그 결과, 아래와 같이 기능변경을 제외한 투입공수($MM_{M2+O}+MM_S$)를 예측하고, 그 정확도를 측정해 보았더니 MMRE=0.12, PRED(25)=1.00로 상당히 개선된 투입공수 모델을 산정할 수 있었다. 아래의 식은 모델 4에 따라서 유지보수 사업대가를 선정하는 방식을 보여준다.

$$Cost_{total}=Cost_{M1}+(MM_{M2+O}+MM_S)*(\text{노임단가})$$

단, $Cost_{M1}=A*(FP_{M1})^B$

$MM_{M2+O}=0.0078*(FP_{Dev})^{0.8797}$

$MM_S=15.919+0.0005*FP_{Dev}$

여기에서 $Cost_{M1}$는 계약 당사자들 간에 해당 연도에 수행할 기능변경 규모(FP_{M1})를 측정하여 대가를 산정하게 된다(우리는 실험적으로 A=0.2872, B=0.9238을 얻었다). 그리고 나머지 비용은 대상 시스템의 규모(FP_{Dev})를 이용하여 유지보수 관리에 소요될 투입공수(MM)를 추정한 다음 노임단가를 곱하여 대가를 산정하게 된다. 이 때 투입공수는 {비기능변경, 애플리케이션 운영}을 포함하는 MM_{M2+O}인 멱함수 회귀모형 $0.0078*(FP_{Dev})^{0.8797}$와 사용자지원에 해당되는 MM_S인 선형 회귀모형 $15.919+0.0005*FP_{Dev}$으로 합산된다. 이러한 회귀모형은 소규모 데이터를 대상으로 실험한 것이므로, 더 많은 데이터를 대상으로 확인할 필요는 여전히 남아 있다.

7. 결 론

본 연구에서는 유지보수의 생산성에 기반하여 소프트웨어 유지보수 대가를 산정하는 모델을 제안하였다. 이러한 대가산정을 위하여 먼저 유지보수 활동을 업무 특성별로 기능변경 활동과 비기능변경 활동, 사용자지원 활동, 애플리케이션 운영으로 구분하였다. 기능변경 활동은 발주자의 추가적인 요구에 의하여 비즈니스 기능을 변경하거나 추가하는 활동으로 매우 유동적이며 계약 의존적이다. 반면에 비기능변경 활동은 오류수정, 환경적응, 예방조치 등과 같이 업무의 규모가 상대적으로 비유동적이다. 또한, 나머지 활동들은 대상 시스템의 규모에 따라 매우 의존적인 관계에 있다. 활동규모를 중심으로 산정된 대가는 다시 유지보수 대상 시스템의 특징과 유지보수 업무의 성격에 따라 적절히 보정된다. 본 연구에서 제시된 모델은 업계에서 수행된 유지보수 프로젝트의 규모와 대가에 대한 데이터를 기반으로 수립, 검증하였다. 이러한 소프트웨어 유지보수 대가산정 방법은 유지보수 활동의 생산성을 바탕으로 이질적인 유지보수 활동들의 전체 대가를 효과적으로 측정할 수 있다. 뿐

만 아니라, 발주자와 수주자 간에 발생할 수 있는 추가적인 비즈니스 요구에 대해서도 유지보수 활동의 규모를 적절하게 반영하여 타당한 대가를 산정할 수 있도록 지원한다. 향후에 유지보수 비용자료 수집이 광범위하게 이루어지고 비용자료의 신뢰성이 확보되면 유지보수 활동의 특성별 인자를 감안하는 보정계수에 대한 연구가 이루어져야 한다.

참고문헌

Boehm, B.W. *et al.(2000)*, Software Cost Estimation with COCOMO II, Prentice-Hall, New Jersey.

Chapin, N., J.E. Hale, K.M. Khan, J.F. Ramil and W. Tan(2001), Types of software evolution and software maintenance, *Journal of Software Maintenance Evolution*.: Research and Practice. 13, 3-30.

IEEE Std 1219(1998), IEEE Standard for Software Maintenance.

IFPUG(2004), Function Point Counting Practices Manual 4.2.

ISBSG(2002), The Software Metrics Compendium.

ISO/IEC 12207(1995), Software Life Cycle Processes.

ISO/IEC 14764(1999), Software Engineering- Software Maintenance.

Longstreet, D.(1990), Software Maintenance and Computers, IEEE Computer Society Press, California.

Martine, J. and C. McClure(1983), Software Maintenance: the Problem and its Solutions, Prentice-Hall, New Jersey.

Ministry of Information and Communication(MIC)(2004), A Guideline of Software Cost Estimation.

National Computerization Agency(NCA)(2002), A Study on the Improvement of Software Maintenance Cost Estimation Guidelines, NCA Report.

NESMA(2001), Function Point Analysis for Software Enhancement.

Sommerville, I.(1996), Software Engineering. Addison-Wesley, England.

UKSMA(2001), Measuring Software Maintenance and Support.

Vliet, H.(1993), Software engineering: principles and practice, Wiley, New York.

배 준 수

서울대학교 산업공학과 학사

서울대학교 산업공학과 석사

서울대학교 산업공학과 박사

현재: 전북대학교 산업정보시스템공학과 전임강사

관심분야: BPM, Workflow, 전자상거래, 소프트웨어산업 정책