소프트웨어 개발 비용을 추정하기 위한 사용사례 점수 기반 모델

박 주 석† · 정 기 원††

요 약

객체지향 개발 방법론을 적용하는 소프트웨어 개발 프로젝트에서 개발 노력 추정 기법으로 사용사례점수(UCP, Use Case Point)에 대한 연구가 계속되고 있다. 기존의 연구는 기술적 요인과 환경적 요인을 적용한 AUCP(Adjusted Use Case Point)에 상수를 곱하여 개발 노력을 계산하는 선형모델을 제시하고 있으나, AUCP와 UUCP(Unadjusted Use Case Point)를 이용하여 개발노력을 추정하는 통계적인 모델은 제시되지 않고 있다. 소프트웨어 규모가 증가함에 따라 개발 기간이 기하급수적으로 증가하는 선형 회귀모델이 부적합 하다는 사실과 UCP 계산과정에서 TCF(Technical Complexity Factor)와 EF(Environmental Factor)를 적용에 따른 FP(Function Point) 오차 발생 문제점을 확인하였다. 이 논문은 사용사례점수를 기반으로 하여 기존 연구의 문제점인 TCF와 EF를 고려하지 않고 직접 UUCP로 부터 개발 노력을 추정할 수 있는 선형, 로그형, 다항식, 거듭제곱 및 지수함수 회귀모델의 성능을 평가한 결과, 가장 적합한 모델로 지수형태의 비선형회귀모델을 도출하였다.

A UCP-based Model to Estimate the Software Development Cost

Juseok Park† · Kiwon Chong††

ABSTRACT

In the software development project applying object-oriented development methodology, the research on the UCP(Use Case Point) as a method to estimate development effort is being carried on. The existing research proposes the linear model calculating the development effort that multiplies an invariant on AUCP(Adjusted Use Case Point) which applied technical and environmental factors. However, the statistical model that estimates the development effort using AUCP and UUCP(Unadjusted Use Case Point) is not being studied. The irrelevant relationship of the linear regression model, whose development period is increasing tremendously as the software size increases, is confirmed. Moreover, during the UCP calculating process, there can be errors in FP by applying the TCF(Technical Complexity Factor) and EF(Environmental Factor). This paper presents a non-linear regression model, that does not consider the TCF and EF, and that estimate the development effort from UUCP directly by utilizing the exponential function. An exponential function is selected among the linear, logarithm, polynomial, power, and exponential model via statistical evaluations of the models mentioned above.

키워드 : 기능점수(Function Point), 완전기능점수(Full Function Point), 사용사례 점수(Use Case Point), 개발비용(Development Cost), 회귀모델(Regression Model)

1. 서 론

소프트웨어 비용(Cost) 추정은 소프트웨어 시스템을 구축하는데 소요되는 시간인 노력(Effort)의 양을 예측하는 과정이다[1]. 소프트웨어 개발비용 추정 분야는 20년 전부터 관심의 대상이 되어 왔으며, 비용에 영향을 미치는 인자(Cost Factor)에 기반을 두고 개발비용을 설명하기 위한 많은 방법들이 연구되고 실제 적용되고 있다. 그러나 이 분야에 대한 집중적인 연구 활동에도 불구하고 현재까지도 일반화된 결론을 유도하지 못하고 있는 실정이다[2, 3].

소프트웨어 규모(Size)에 대한 정량화는 소프트웨어 프로젝트에 소요되는 노력, 비용과 개발일정을 적절히 추정하기 위한 핵심적인 인자 중 하나로 일반적으로 인식되고 있다[4]. 소프트웨어 규모는 길이(Length), 기능성(Functionality)과 복잡도(Complexity)로 정의될 수 있으며[4], 라인 수(Lines Of Code, LOC)와 기능점수(Function Point, FP)가 일반적으로 사용되고 있다[1, 4]. 라인 수는 길이에 기반을 두고 있어 사용되는 프로그래밍 언어에 따라 다른 크기로 정량화되며, 코딩이 완료된 이후에야 정확한 크기를 측정할 수 있는 취약점을 갖고 있다[5]. Albrecht[6]는 사용자에게 제공되는 소프트웨어의 기능성과 복잡도에 기반을 두고 규모를

※ 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음.
† 준 회 원 : 숭실대학교 대학원 컴퓨터학과
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
논문접수 : 2003년 7월 24일, 심사완료 : 2003년 9월 15일

정량화하는 기능점수 방법을 처음으로 제안하였다. 이 방법은 데이터 관리 위주인 관리정보시스템(Management Information System, MIS) 소프트웨어에 적합하도록 설계되어, 1990년대 들어 다른 소프트웨어 분야에서는 적용하기가 어렵다는 취약점을 나타내었다[6, 7]. 1998년 COSMIC(Common Software Management International Consortium)이 설립되어 기능점수 기법을 제어 위주인 실시간 시스템(Real-time System)과 내장형 시스템(Embedded System)으로 적용 범위를 확장하였으며, 이를 완전 기능점수(Full Function Point, FFP)라 칭하였다. 그러나 이 방법은 복잡한 수학계산으로 구성된 알고리즘 위주의 소프트웨어 기능성을 측정하도록 설계되지 못하였다[7, 8].

기능점수나 완전 기능점수는 기능성에 기반을 두고 있어 트랜잭션 처리나 파일 형태에 적합하며, 자료흐름도(Data Flow Diagram), 계층적 프로세스 모델 또는 데이터베이스 구조와 같은 전통적인 구조적 설계기법으로부터 유도된다[4]. 따라서, 객체지향 분석과 설계에 기능점수를 적용하는 방안에 대한 연구가 진행되고 있다[9, 10]. 기능점수를 객체지향 개발방법에 적용하기 위해서는 객체지향 개념으로 분석된 요구사항 분석단계 결과물들을 기능점수의 구성요소인 입력, 출력, 조회, 인터페이스와 파일로 변환시키는 추가적인 작업이 필요하다. 대부분의 소프트웨어 개발의 실패 원인이 개발 초기에 사용자의 요구사항 도출 어려움에 기인하여 이를 해결하려는 연구가 계속되었다. 1992년 Jacobson[11]에 의해 요구사항 도출방법으로 사용사례(Use Case)가 제안되었고, 1998년 UML(Unified Modeling Language)에 사용사례도(Use Case Diagram)가 포함된 이후 소프트웨어 산업계에서는 사용사례-구동(Use Case-driven), UML 기반의 단일화된 프로세스(Unified Process)로 객체지향 소프트웨어를 개발하고 있다[4]. 2000년대 들어 새로운 패러다임으로 등장한 컴포넌트 기반 소프트웨어 개발(Component-based Development, CBD)에도 이 기법들이 적용되고 있다[12].

1993년 Karner[13]는 Albrecht[6]의 기능점수 계산방법을 수정(Modify)하여 사용사례 점수(Use Case Point, UCP)를 계산하는 방법을 제시하였다. 이후에도 사용사례 점수에 대한 연구는 계속되고 있으나 사용사례 점수에 기반하여 개발비용을 추정할 수 있는 모델 연구는 미미한 수준이다. 따라서 본 논문은 소프트웨어 개발에 소요되는 비용을 추정함에 있어, 구조적 개발방법론에 적합한 기존의 라인 수와 기능점수를 지양하고 객체지향 개발방법론에 적합한 사용사례 점수로 측정된 소프트웨어 규모에 따른 개발비용을 추정할 수 있는 통계적 모델을 제시한다.

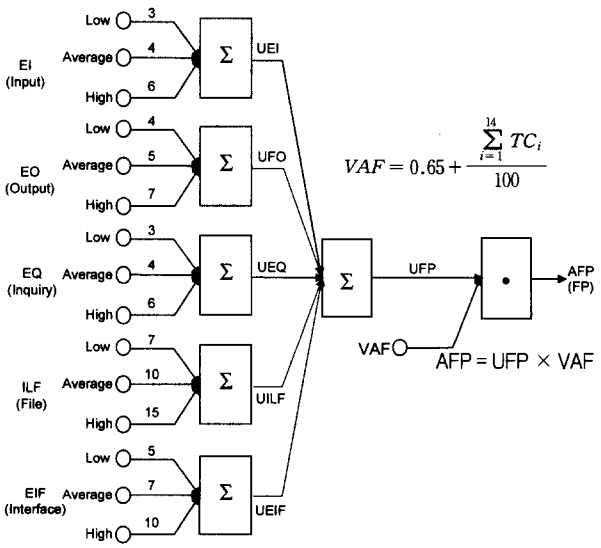
2장에서는 소프트웨어 규모 추정 기법에 대해 간략히 제시하고, 3장에서는 사용사례 점수 기반 소프트웨어 비용추정과 관련된 기존의 연구결과를 살펴보고 문제점을 제시한다. 4장에서는 사용사례 점수 기반 소프트웨어 비용추정을 위한 통계적 회귀모델을 제시하고 모델의 적합성을 검증한다.

2. 소프트웨어 규모추정 기법

사용사례 점수는 기능점수를 수정하여 유도된 관계로 사용사례 점수를 이해하려면 기능점수를 먼저 이해하는 것이 필요하다. 본 장에서는 기능점수와 사용사례 점수를 계산하는 방법을 간략히 살펴본다.

2.1 기능점수

사용자에게 제공되는 응용 프로그램의 규모를 계산 가능한 기능성으로 측정하기 위해, 응용 프로그램을 데이터 기능(Data Function)과 처리 기능(Transaction Function)으로 구분한다. 데이터 기능은 내부와 외부 데이터 요구사항에 일치하도록 사용자에게 제공되는 것으로 화일(Internal Logical Files, ILF)과 인터페이스(External Interface File, EIF)가 있다. 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. 기능점수 계산 과정은 (그림 1)에 간략히 표현하였으며, 다음 순서로 계산된다[14].



(그림 1) 기능점수 계산과정

Step 1 : 5가지 기능요소 형태(ILF, EIF, EI, EO 또는 EQ)를 파악한다.

Step 2 : 기능요소 형태 각각에 대해 <표 1>의 기준에 따라 기능적 복잡도 수준을 결정한다.

〈표 1〉 복잡도 수준 계산

(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

RET(Record Element Type)는 파일 또는 인터페이스의 데이터 요소 서브그룹, DET(Data Element Type)는 필드, FTR(File Types Referenced)는 읽히거나 유지되는 파일을 의미한다.

Step 3 : <표 2>의 가중치를 참조하여 <표 3>으로 UFP (Unadjusted FP)를 계산한다.

〈표 2〉 가중치

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

〈표 3〉 UFP 계산

기능요소 형태	기능 복잡도			Total (① + ② + ③)
	Low(①)	Average(②)	High(③)	
EI	_ × 3 = _	_ × 4 = _	_ × 6 = _	
EO	_ × 4 = _	_ × 5 = _	_ × 7 = _	
EQ	_ × 3 = _	_ × 4 = _	_ × 6 = _	
ILF	_ × 7 = _	_ × 10 = _	_ × 15 = _	
EIF	_ × 5 = _	_ × 7 = _	_ × 10 = _	
UFP(Unadjusted Function Points)				

Step 4 : 사용자에게 제공되는 응용 프로그램의 일반적인 시스템 속성 (General System Characteristics, GSC)을 <표 4>에 의거 평가한 후, 식 (1)의 VAF(Value Adjustment Factor)를 계산한다.

$$VAF = (TDI \cdot 0.01) + 0.65 \quad (1)$$

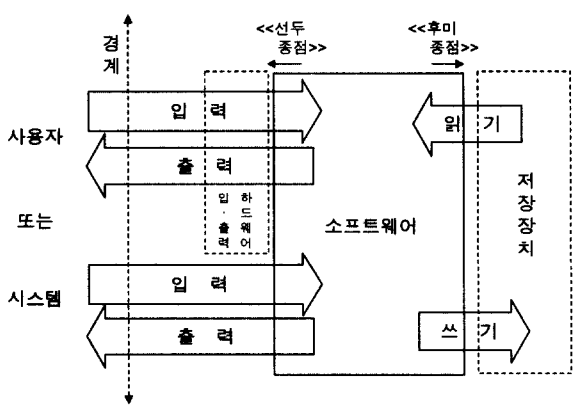
〈표 4〉 TDI 계산

GSC	DI(Degree of Influence)	Remark
데이터 통신		[DI] 0 : 영향 없음 1 : 우발적 영향 2 : 보통의 영향 3 : 평균적인 영향 4 : 중요한 영향 5 : 완전히 강한 영향
분산 데이터 처리		
성 능		
가중된 사용 형상		
처리율		
온라인 데이터 입력		
사용자 능률		
온라인 갱신		
복잡한 처리		
재사용성		
설치 용이성		
운영 용이성		
다중 설치운영		
변경 용이성		
TDI(Total Degree of Influence) = $\sum_{i=1}^{14} DI_i$		

Step 5 : 식 (2)에 의거 소프트웨어 프로젝트에 대한 최종 조절된 기능점수(Adjusted FP, AFP)를 계산한다. AFP를 일반적으로 기능점수(FP)라 칭한다.

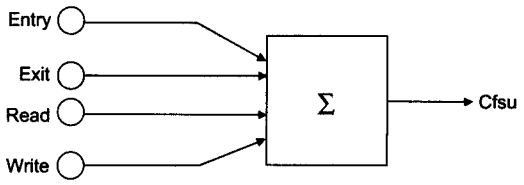
$$AFP = VAF \cdot UFP \quad (2)$$

완전 기능점수는 (그림 2)와 같이 소프트웨어 시스템을 4종류의 데이터 이동 서브프로세스로 구별한다[7, 8]. 소프트웨어 관점에서 볼 때, 입력(ENTRIES)와 출력(EXITS)는 사용자와 주고받는 데이터 속성이며, 읽기(READS)와 쓰기(WRITES)는 저장장치와 주고받는 데이터 속성이다.



(그림 2) FFP 구성요소

완전 기능점수에 기반한 소프트웨어 규모는 Cfsu(Cosmic Functional Size Unit)로 표기하며, 임의의 i번째 계층에 대한 Cfsu는 (그림 3)의 계산 과정에 따라 식 (3)과 같이 계산된다[7, 8].

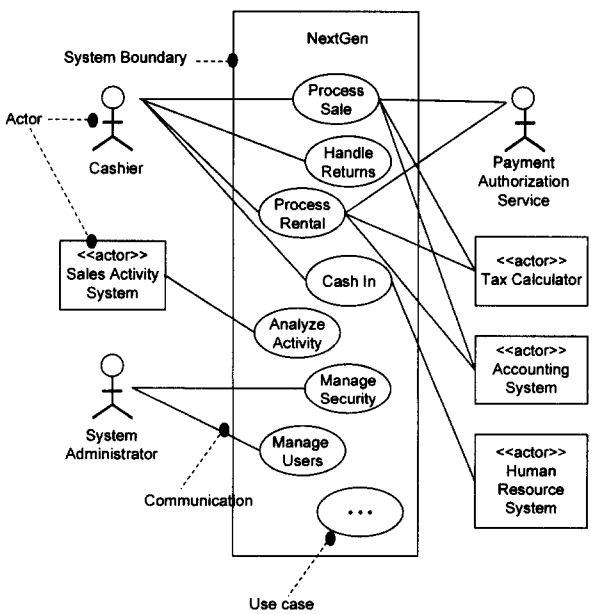


(그림 3) FFP 계산과정

$$Size_{Cfsu}(layer_i) = \sum size(entries_i) + \sum size(exits_i) + \sum size(reads_i) + \sum size(writes_i) \quad (3)$$

2.2 사용사례 점수

사용사례는 사용자가 추구하는 목표(Goals) 또는 필요로 하는 것(Needs)을 충족하기 위해 시스템을 사용하는 이야기(Stories)로 요구사항을 이해하고 기술하는 탁월한 기법으로 알려져 있다[15]. 사용사례는 1992년 Jacobson[11]에 의해 제안되었으며, 객체지향 소프트웨어 개발 전 단계에 적용되고 있다[11, 15]. 사용사례 모델은 사용사례 설명서(Use Case Description)와 사용사례도(Use Case Diagram)로 구성되어 있으며, 일반적으로 불리우는 사용사례는 사용사례 설명서를 의미하며, 사용사례도는 사용사례 설명서를 보충 설명하기 위한 용도로 사용된다[15]. 주 행위자(Primary Actor)의 사용자 목표를 만족시키기 위해 사용사례가 정의되기 때문에 사용사례 모델링은 먼저 시스템의 범위(Boundary)를 선택하고, 주 행위자를 식별한 후, 주 행위자의 사용자 목표를 식별하여 사용자 목표를 만족시키는 사용사례를 정의(기술)하는 단계를 거친다. (그림 4)는 슈퍼마켓의 판매관리 시스템인 NextGen에 대한 사용사례도의 일부분을 보여주고 있다.

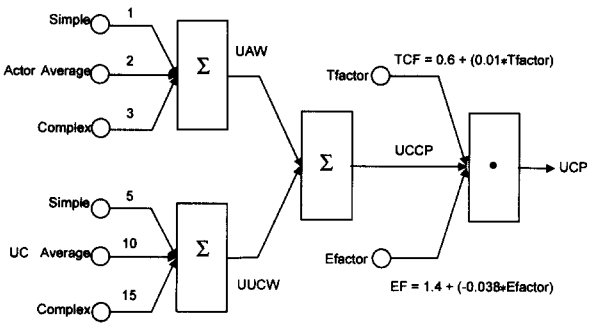


(그림 4) NextGen 시스템의 Use case 배경도

사용사례 점수를 계산하는 방법은 (그림 5)에 제시하였으며, 다음 순서로 계산된다[4, 13, 15].

Step 1 : <표 5>에 의해 액터 복잡도인 UAW(Unadjusted Actor Weights)를 계산한다.

Step 2 : <표 6>에 따라 사용사례 복잡도인 UUCW(Unadjusted Use Case Weights)를 계산한다.



(그림 5) UCP 계산과정

〈표 5〉 UAW 계산

Actor Type	Description	Weight Factor (①)	Number of Actors (②)	UAW_i (①×②)
Simple	응용 프로그래밍 인터페이스(API)를 통해 인터페이스 된 타 시스템	1		
Average	대화식 또는 TCP/IP 등의 프로토콜 을 통해 상호작용하는 타 시스템	2		
Complex	GUI 또는 Web 페이지를 통해 상호작용하는 사용자	3		
UAW(Unadjusted Actor Weights) = $\sum_{i=1}^{3} UAW_i$				

〈표 6〉 UUCW 계산

Use Case 형태	방법		Weight Factor (①)	Number of Use Cases (②)	$UUCW_i$ (①×②)
	Number of Transaction (Key Scenario)	Number of Classes			
Simple	≤ 3	< 5	5		
Average	4~7	5~10	10		
Complex	≥ 8	> 10	15		
UUCW(Unadjusted Use Case Weights) = $\sum_{i=1}^{3} UUCW_i$					

Step 3 : 식 (4)의 UUCP(Unadjusted Use Case Points)를 계산한다.

$$UUCP = UAW + UUCW \quad (4)$$

Step 4 : <표 7>의 TFactor 계산 후 식 (5)에 의거하여 생산성에 영향을 미치는 기술적 복잡도 요인(TCF : Technical Complexity Factor)을 계산한다.

$$TCF = 0.6 + (0.01 \cdot TFactor) \quad (5)$$

Step 5 : <표 8>의 Efactor 계산 후 식 (6)에 의거 생산성에 영향을 미치는 환경적 복잡도 요인(Environmental Complexity Factor, EF)를 계산한다.

$$EF = 1.4 + (-0.03 \cdot EFactor) \quad (6)$$

〈표 7〉 TFactor 계산

기술적 요 인	Description	Weight Factor(①)	Assigned Value(②)	TF_i (①×②)
T1	분산시스템	2	0~5	
T2	응답시간 또는 처리성능	1		
T3	사용자 능력	1		
T4	내부처리 복잡도	1		
T5	코드 재사용성	1		
T6	설치 용이성	0.5		
T7	사용 용이성	0.5		
T8	휴대용	2		
T9	변경 용이성	1		
T10	병렬처리	1		
T11	특수 보안	1		
T12	제 3자 집적 접근성	1		
T13	특별한 사용자 교육시설	1		
TFactor(Total Technical Factor) = $\sum_{i=1}^{13} TF_i$				

〈표 8〉 EFactor 계산

환경적 요인	Description	Weight Factor(①)	Assigned Value(②)	EF_i (①×②)
E1	RUP 친숙도	1.5	0~5	
E2	해당분야 경험	0.5		
E3	객체지향 경험	1		
E4	분석책임자 능력	0.5		
E5	동기부여	1		
E6	요구사항 안정 정도	2		
E7	비상근 개발요원	-1		
E8	프로그램 언어 어려움	-1		
EFactor(Total Environmental Factor) = $\sum_{i=1}^{8} EF_i$				

Step 6 : 식 (7)의 조절된 사용사례 점수(Adjusted Use Case Point, AUCP)를 계산한다. AUCP를 UCP라 칭한다.

$$AUCP = UUCP \times TCF \times EF \quad (7)$$

TFactor와 EFactor에서 적용되는 가중치 할당 값은 기능점수의 DI 계산에 적용된 기준과 동일하다.

3. 사용사례 점수 기반 개발노력 추정 관련연구

기능점수 방법을 적용하는 실무자들은 UFP가 AFP와 같이 정확성을 가지고 있기 때문에 VAF 값을 무시하고 있다.[4] 또한, Kitchenham과 Känsälä [17]도 VAF 값이 개발노력의 추정 능력을 향상시키지 못함을 보였다. 이에 따라 Ribu[4]도 식 (7)의 AUCP 계산에서 TCF를 제외시켜도 개발노력 (E)를 추정하는데는 영향이 없음을 제시하였다.

UCP를 이용하여 LOE(Level Of Effort)를 얻기 위해 식 (8)의 형태에 대한 연구가 다양하게 진행되었다.

$$E = AUCP \times \text{쓰임새 점수 당 개발 소요시간} \quad (8)$$

사용사례점수 당 개발 소요시간(시간/UCP)는 개발조직의 생산성에 의존한다[16, 18]. Banerjee[16]는 실제 적용 경험에 의하면 $15 \le \text{시간}/UCP \le 30$의 범위를 갖고 있다고 제안하였으며, Probasco[18]는 20시간부터 시작하여 개발조직의 생산성에 따라 조절해야 함을 제시하였다. Karner[13]는 20시간/UCP를 적용하였으며, Schneider와 Winter는 $EF \le 2$인 경우 20시간/UCP를, $3 \le EF \le 4$인 경우 28시간/UCP, $EF \ge 5$인 경우 위험요소가 허용이 불가능할 정도로 크므로 EF를 조절하는 것이 필요함을 제시하였다[16]. Ribu[4]는 36시간/UCP를 적용하는 특별한 경우도 경험하였으며, 학생들이 수행한 프로젝트에는 10시간/UCP를 적용하였다. Sun 사는 30시간/UCP, 어떤 업체에서는 중간값을 취함을 경험하였다[16].

사용사례 점수에 기반을 둔 개발노력 추정 기존 연구들은 다음과 같은 문제점들을 내포하고 있다.

① 기존의 연구들은 AUCP에 상수를 곱하여 개발노력을 계산하였다. 즉, 개발노력은 AUCP에 선형적인 관계를 갖는 1차 함수($E = a \cdot AUCP$)로 프로젝트의 규모에 상관없이 UCP 당 일정한 시간이 투입됨을 의미하나 이는 현실적으로 적합하지 않다. 왜냐하면 프로젝트의 규모가 커지면 프로그램의 복잡도가 기하급수적으로 증가하는 경향을 띠고 있어 개발에 투입되는 시간도 프로젝트 규모에 따라 기하급수적으로 증가하는 비선형적인 관계를 나타낸다. 따라서, 선형회귀모델 보다는 비선형 회귀모델이 보다 적합할 수 있다.

② UCP 계산 과정에서 VAF와 동일한 개념인 TCF는 최종적으로 계산되는 UCP에 ±40%의 편차를 나타내며, EF는 -30~+40%의 편차를 유발시킬 수 있다. 그러나 기능점수 계산 과정에서 VAF는 최종적으로 계산되는 FP에 대해 ±35%의 오차를 유발시킬 수 있음에도 불구하고 효용성이 없는 것으로 연구되어 AFP 보다는 UFP를 적용하고 있다. 따라서, TCF나 EF를 적용한 AUCP 보다는 UUCP만으로 개발노력을 추정할 수 있을 것이다.

AUCP나 UUCP를 이용하여 개발노력이나 비용을 추정하는 통계적인 모델은 제시되지 않고 있다. 따라서, 4장에서는 사용사례 점수에 기반하여 개발비용을 추정할 수 있는 통계적인 모델을 제시한다.

4. 사용사례 점수 기반 개발비용추정 모델

추정된 개발노력(시간)에 대해 개발업체의 단위 시간당 평균 소요 비용을 곱하면 개발비용이 산출된다. 이와 같은 이유로 인해 일반적으로 개발노력 추정 모델을 개발비용 추정 모델이라 부른다[3]. 이후부터는 개발노력을 추정하는 것을

〈표 9〉 사용사례 점수 관련 데이터

프 로 젝 트		Actor				Use Case			
		Simple	Medium	Complex	UAW	Simple	Medium	Complex	UUCW
S. Nageswaren [19]					55				64
Ribu[4]	Project B-Subsystem 1	1	3	2	13	10	7	2	150
	Project B-Subsystem 2	0	4	1	11	12	9	0	150
	Project B-Subsystem 3	0	5	0	10	2	4	6	145
	Project B-Subsystem 4	1	3	2	13	0	8	2	110
	1-Q	0	1	2	8	0	6	0	60
	2-S	0	0	3	9	6	1	0	40
	3-S	0	0	3	9	4	2	0	40
	4-Q	1	0	2	7	2	3	1	55
	5-Q	1	0	2	7	0	4	0	40
	6-S	1	0	2	7	2	7	0	80
	7-Q	1	1	2	9	1	2	0	25
	8-S	0	1	3	11	2	3	1	55
	9-S	1	0	2	7	1	2	1	40
	10-Q	1	1	2	9	4	4	0	60
	Project A	0	0	2	6	18	41	4	560

프 로 젝 트		UUCP	TC Factor	E Factor	TCF	EF	AUCP	Staff-Hour per UCP	Effort
S. Nageswaren[19]		119	87	13	1.47	1.01	176.68	20	3,120
Ribu[4]	Project B-Subsystem 1	163	43	17.5	1.03	0.88	146.90	28	3,723
	Project B-Subsystem 2	161	43	17.5	1.03	0.88	145.10	28	3,665
	Project B-Subsystem 3	155	43	17.5	1.03	0.88	139.69	28	3,835
	Project B-Subsystem 4	123	43	17.5	1.03	0.88	110.85	28	2,710
	1-Q	68	25	19	0.85	0.83	47.97	10	294
	2-S	49	19	19	0.79	0.83	32.13	10	298
	3-S	49	19	19	0.79	0.83	32.13	10	232
	4-Q	62	25	19	0.85	0.83	43.74	10	371
	5-Q	47	25	19	0.85	0.83	33.16	10	420
	6-S	87	19	19	0.79	0.83	57.05	10	580
	7-Q	34	25	19	0.85	0.83	23.99	10	243
	8-S	66	19	19	0.79	0.83	43.28	10	595
	9-S	47	19	19	0.79	0.83	30.82	10	578
	10-Q	69	25	19	0.85	0.83	48.68	10	490
	Project A	566	44	19	1.04	0.92	541.55	10	10,043

개발비용 추정이라 칭한다. 소프트웨어 비용추정 모델들은 일반적으로 주요 비용인자인 규모와 부수적인 조절인자로 구성되어 있다. 전형적인 비용 추정 모델은 과거 개발된 소프트웨어 프로젝트로부터 수집된 데이터에 대한 회귀분석으로부터 유도되며, 모델의 구조는 식 (9)로 표현된다[15].

$$E = a + b \cdot S^c \tag{9}$$

여기서 E는 개발노력(Person-months 또는 Person-Hours), a, b, c는 경험적으로 유도된 상수이며 S는 주요 비용인자인 규모로서 LOC 또는 FP가 된다.

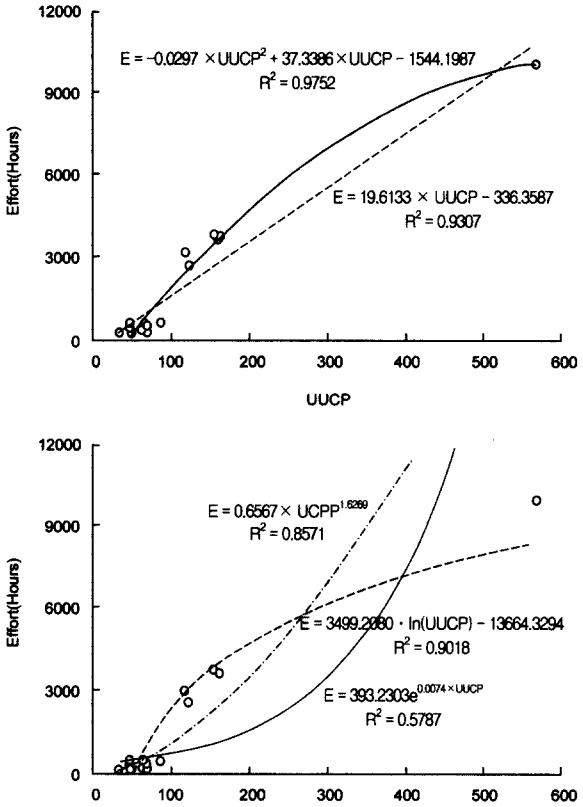
본 제안 모델은 3장에서 제시한 기존 연구의 문제점 ②에 따라 TCF와 EF를 고려하지 않고 직접 UUCP로부터 개발노력을 추정할 수 있는 통계적 모델을 제시한다. 모델 제시를 위해 Ribu[4]와 Nageswaren[19]의 연구결과로부터 <표 9>의 데이터를 획득하였다.

<표 9>에서 Project A는 다른 데이터들에 비해 UUCP와 Effort 값이 월등히 큰 이상점을 갖고 있어 이 데이터를 포함시켜 모델을 유도하면 편향된 결과를 얻을 수 있다. 따라서, 본 논문에서는 Project A 데이터를 포함시킨 모델은 참고용으로 제시한다.

적합한 통계적 회귀모델을 선정하기 위해 결정계수(Coefficient of determination, R^2)[20]를 적용하며, 모델의 성능을 평가하기 위해 Briand et al.[2]이 적용한 MMRE(Mean Magnitude of Relative Error)를 이용한다. 종속변수의 값차의 합으로 나타나며, 총 변동을 설명하는데 있어서 회귀직(E)은 독립변수(UUCP)에 의해 결정되는 부분과 미지의 오선에 의해 설명되는 변동 비율을 결정계수라 하며, 값이 클수록 쓸모 있는 회귀직선이 되지만 좋은 모델로 선정하기 위한 기준은 없는 실정이다. 모델의 정확도(Accuracy)를 평가하는 MMRE는 다음과 같이 측정된다. $RE = \frac{\text{추정치} - \text{실측치}}{\text{실측치}}$, MRE(Magnitude of RE) = ABS(RE), MMRE(Mean MRE) $= 100/n * \sum_{i=1}^{n} MRE$. MMRE가 작은 값이면 평균적으로 좋은 모델임을 알 수 있다. Conte et al.[21]는 $MMRE \leq 0.25$ (25%)이면 개발비용을 예측하는 모델로 채택 가능한 것으로 고려하였다.

4.1 UUCP 기반 개발노력 추정 모델(Project A 포함)

Project A를 포함한 경우에 대해 UUCP에 따른 개발비용 추정모델은 (그림 6)에, 모델의 성능 분석 결과는 <표 10>에 제시되어 있다. Project A를 포함한 경우, 선형, 로그와 지수 함수 형태는 100 이상의 UUCP에 대한 개발노력(시간)을 표현하지 못하고 있으며, 거듭제곱 함수 형태가 가장 적합한 모델로 선정될 수 있다.



(그림 6) 회귀분석 모델(프로젝트 A 포함)

〈표 10〉 개발비용 추정 모델 성능(프로젝트 A포함)

모델형태	모 델	결정계수	MMRE
선 형	$E = 19.6133 \cdot UUCP - 336.3587$	0.9307	73.68
로 그	$E = 3499.2080 \cdot \ln(UUCP) - 13664.3294$	0.9018	127.90
다 항 식	$E = -0.0297 \cdot UUCP^2 + 37.3386 \cdot UUCP - 1544.1987$	0.9752	61.03
거듭제곱	$\boldsymbol{E = 0.6567 \cdot UUCP^{1.6269}}$	**0.8571**	**43.42**
지 수	$E = 393.2303 \cdot e^{0.0074 \cdot UUCP}$	0.5787	70.84

4.2 UUCP 기반 개발노력 추정 모델(Project A 제거)

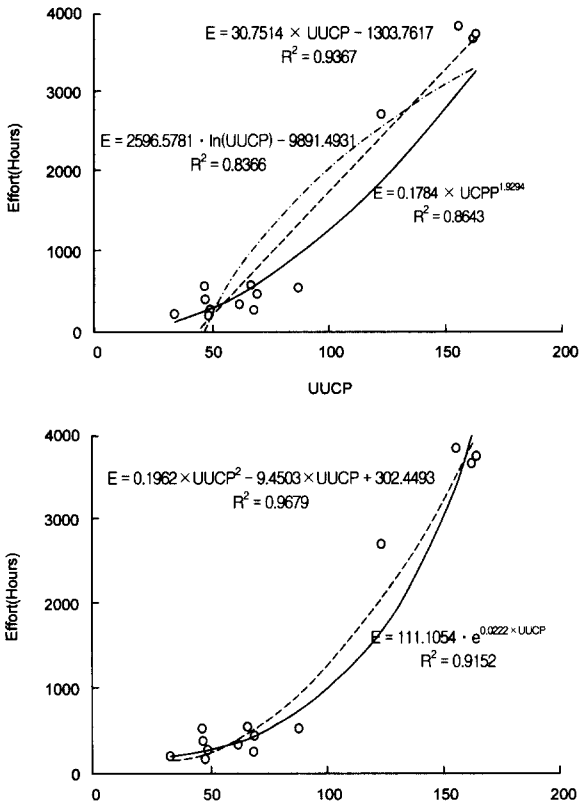
Project A를 제거하고 나머지 데이터들만을 이용하여 UUCP에 따른 개발비용 추정모델은 (그림 7)에, 모델의 성능 분석 결과는 <표 11>에 제시되어 있다.

Project A를 제거시 선형과 로그 함수 형태는 40 이하의 UUCP를 가진 프로젝트 규모에 대해서는 음수 값을 가지는 개발시간을 추정하여 현실성이 없다. 지수와 다항식 함수

형태의 모델이 전반적으로 모든 데이터를 가장 잘 표현하는 모델이라 할 수 있으며, 지수함수 형태의 모델이 MMRE가 가장 적고 Conte et al.[21]의 개발노력 예측 모델 선정조건을 거의 만족시키고 있다. 따라서, UUCP를 이용해 개발노력을 추정하는 모델로 $E = 111.1054 \cdot e^{0.0222 \cdot UUCP}$를 제안한다.

기존 연구들이 수행한 조절된 사용사례 점수에 상수를 곱한 형태의 모델들과 제안된 모델의 성능을 비교하여 보자. 기존 연구들은 $10 \le$ 시간/$AUCP \le 30$으로 다양한 결과를 보였다. 그러나 일반적으로 $15 \le$ 시간/$UCP \le 30$의 범위를 갖고 있으므로 AUCP당 개발시간을 15, 20과 30을 적용하여 <표 9>의 데이터들에 대한 MMRE를 계산하면 각각 48.93, 72.87과 142.95를 얻을 수 있다. 이는 제안된 모델의 MMRE 성능인 25.85에 비해 월등히 좋지 않은 결과를 나타내며, 기존 연구들을 이용할 경우 AUCP당 개발시간을 얼마로 결정할 것인가에 따라 개발비용에 큰 편차를 나타낼 수 있다.

개발될 소프트웨어 시스템의 UUCP를 계산한 후, 제안된 모델을 이용하여 개발노력을 추정하고 개발업체의 시간당 평균 소요 비용을 곱하면 개발에 소요되는 직접비용을 구할 수 있다.



(그림 7) 회귀분석 모델(프로젝트 A 제거)

〈표 11〉 개발비용 추정 모델 성능(프로젝트 A 제거)

모델 형태	모 델	결정계수	MMRE
선 형	$E = 30.7514 \cdot UUCP - 1303.7617$	0.9367	59.45
로 그	$E = 2596.5781 \cdot \ln(UUCP) - 9891.4931$	0.8366	94.96
다항식	$E = 0.1962 \cdot UUCP^2 - 9.4503 \cdot UUCP + 302.4493$	0.9679	28.50
거듭제곱	$E = 0.1784 \cdot UUCP^{1.9294}$	0.8643	34.90
지 수	$\boldsymbol{E = 111.1054 \cdot e^{0.0222 \cdot UUCP}}$	**0.9152**	**25.85**

지금까지는 주어진 데이터에 적합한 모델을 선정하기 위한 모델의 추정(Estimation) 능력과 관련된 연구를 수행하였다. 다음으로 획득된 모델이 미지의 데이터에도 적합한지 여부를 판단할 수 있는 예측(Prediction) 능력을 판단하여 보자. 모델의 예측능력을 판단하는데 Conte et al.[21]의 Pred (0.25) 척도를 많이 적용하고 있다. Pred(0.25)는 소프트웨어의 규모에 따라 주어진 모델로 측정된 비용의 값이 실제 값의 25% 내에 속하는 비율을 의미한다. Pred(0.25)가 0.50 이라면 산정 값의 50%가 실제 값의 50% 내에 속한다는 것이다. Conte et al.[21]은 Pred(0.25)가 0.75 이상일 경우 뛰어난 비용산정 예측모델이 될 수 있음을 제시하였다. 제안된 모델들의 Pred(0.25)를 분석한 결과는 <표 12>에 제시되어 있다.

〈표 12〉 Pred(0.25)에 따른 모델의 예측 성능

모델 형태	모 델	
	프로젝트 A 제거시	프로젝트 A 포함시
선 형	0.4667	0.3125
로 그	0.4000	0.3750
다항식	0.5333	**0.4375**
거듭제곱	0.3333	0.2500
지 수	**0.6000**	0.1250

지금까지 제시된 비용산정 모델들 조차도 Conte et al. [21]의 Pred(0.25) 기준을 만족하지 못하는 경우가 일반적이다. 본 논문에서 제안된 모델들도 비용 예측을 위한 이 기준을 만족하지는 못하지만 프로젝트 A 제거시에는 지수 모델이 60%를, 프로젝트 A 포함시에는 다항식 모델이 43.75%를 예측할 수 있는 능력을 갖고 있다.

5. 결론 및 향후 연구과제

소프트웨어 개발에 적용되는 방법론과 언어는 보다 좋은 품질, 보다 빠른 개발과 보다 저렴한 개발비용을 투입하기

위한 방향으로 급격한 변화를 거듭하고 있다. 현재는 구조적 개발 방법론보다는 객체지향 방법론이 소프트웨어 산업계를 주도하고 있다. 소프트웨어 비용추정 분야도 이러한 변화에 적응하기 위해서는 기존의 구조적 개발방법론에 적합한 라인 수와 기능점수 기반의 추정모델을 적용하기보다는 객체지향 개발방법론에 적합한 사용사례 점수 기반의 비용추정 모델로의 전환이 필요하다. 그러나 이 분야에 대한 연구가 거의 수행되지 않고 있다. 이러한 추세에 부응하고자 본 논문은 사용사례 점수에 기반한 소프트웨어 비용추정 모델을 제시하였다.

본 논문에서는 사용사례 점수 계산 방법을 살펴보고, 사용사례 점수를 기반으로 하여 개발비용을 추정하는 기존 의 선형 모델의 문제점을 제기하였다. 제안된 모델은 최종적으로 계산된 조절된 사용사례 점수에 기반하지 않고, 조절이 안 된 사용사례 점수에 기반하여 최적의 개발비용을 추정하는 모델을 제시하였다. 모델의 성능을 평가한 결과 지수형태의 비선형 회귀모델($E = a \cdot e^{b \cdot UUCP}$)이 가장 적합한 모델로 선정되었다.

이 모델은 개발 노력이 4000 미만인 프로젝트에 적용 가능하며 4000 이상인 프로젝트에 적용 가능한 일반화된 모델을 얻기 위해서는 다양한 환경에서 다양한 개발업체들로부터 획득된 충분한 양의 데이터들로부터 모델이 유도되어야만 한다. 그러나 사용사례 점수를 적용한 사례가 극소수에 불과하여 모델 제시에 적용된 데이터의 양이 불충분한 취약점은 갖고 있다. 그러므로 추후에는 사용사례 점수 기반의 비용추정 모델에 대한 보다 심도있는 연구와 더불어 다양한 데이터 수집으로 일반화된 모델을 제시하는 연구가 수행될 것이다.

참 고 문 헌

[1] K. Johnson, "Software Cost Estimation : Metrics and Models," Department of Computer Science University of Calgary, Albreta, Canada, http://sern.ucalgary.ca/courses/seng/621/W98/johnsonk/cost.htm, 1998.

[2] L. C. Briand, K. E. Elmam, D. Surmann, I. Wieczork and K. D. Maxwell, "An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques," International Software Engineering Research Network, Technical Report, ISERN-98-27, 1998.

[3] L. C. Briand and I. Wieczorek, "Resource Estimation in Software Engineering," International Software Engineering Research Network, Technical Report, ISERN 00-05, 2000.

[4] K. Ribu, "Estimating Object-oriented Software Projects with Use Cases," University of Oslo Department of Informatics, Master of Science Thesis, 2001.

[5] 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.

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

[7] C. Symons, "COSMIC-FFP Measurement Manual, Version 2.1," Common Software Measurement International Consortium, 2001.

[8] C. Symons, "COSMIC-FFP Measurement Manual, Version 2.2 (The COSMIC Implementation Guide for ISO/IEC 19761 : 2003)," Common Software Measurement International Consortium, 2003.

[9] J. Kammelar, "A Sizing Approach for OO-environments," IQUIP Informatica B. V. Netherlands, 4th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, 2000.

[10] T. Fetcke, A. Abran and T-H. Nguyen, "Mapping the OO-Jacobson Approach into Function Point Analysis," Université du Québec á Montréal Software Engineering Management Research Laboratory, IEEE Proceeding of TOOLS-23, 1998.

[11] I. Jacobson, M. Christerson, et al., "Object-oriented Software Engineering. A Use Case Driven Approach," Addison-Wesley, 1992.

[12] M. Carbune and G. Santucci, "Fast & Serious : A UML based Metric for Effort Estimation," Universita' degli studi di Roma "La Sapieuza," http://alarcos.inf-cr.uclm.es/qaoose2002/docs/QAOOSE-Car-San.pdf, 2002.

[13] G. Karner, "Metrics for Objectory," Diploma Thesis, University of Linköping, Sweden, No. LiTH-IDA-Ex-934421, 1993.

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

[15] C. Larman, "Applying UML and Patterns. An Introduction to Object-oriented Analysis and Design and the Unified

Process," Prentice-Hall, 2002.

[16] G. Banerjee, "Use Case Points-An Estimation Approach," http://java.isawix.com/whitepapers/1035194512861.pdf, 2001.

[17] B. Kitchenham and K. Känsälä, "Inter-item Correlation Among Function Points," National Computing Centre Ltd, UK and VTT, Finland, 1997.

[18] L. Probasco, "Dear Dr. Use Case : What About Function Points and Use Cases," http://www.therationaledge.com/content/aug_02/t_drUseCase_lp.jsp, Rational Software Canada, 2002.

[19] S. Nageswaren, "Test Effort Estimation Using Use Case Points," Quality Week 2001, San Francisco, California, USA, 2001.

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

[21] S. Conte, H. E. Dunsmore and V. Y. Shen, "Software Engineering Metrics and Models," Benjamin/Cummings., 1986.

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)