소프트웨어의 품질을 고려한 비용 평가 모델의 제안과 평가

이 용 근† 양 해 술††

요 약

최근 소프트웨어 적용분야가 확대됨에 따라 소프트웨어가 차지하는 비중이 점차 높아지고 개발 비용에 대한 중요성이 증대되고 있다. 그러나 종래의 개발비용 평가 모델은 주로 소프트웨어의 기능면에서 평가한 것이 대부분이었다. 본 논문에서는 소프트웨어의 가치를 기능면뿐만 아니라 품질면에서도 평가할 수 있는 소프트웨어 개발비용 평가 모델 COSMOS-Q(COSt MOdel for Subcontract-Quality)를 제안한다. 제안 모델의 목적은 소프트웨어 발주자가 발주 명세 정보만을 기본으로 정확한 소프트웨어의 비용을 견적할 수 있도록 하는 것이다. 특히, 본 연구에서는 ISO/SC7의 소프트웨어 품질특성에 관한 검토 결과를 참고로 비용에 변동을 부여한 품질특성 요인의 추출 및 발주조건으로 지정가능한 각 요인의 평가척도를 설정하여 품질을 고려한 비용 평가 모델을 제안하고 그 타당성을 평가하였다.

Proposal and Evaluation of a Cost Estimation Model Considering Software Quality

Ryong Geun Rhee† and Hae Sool Yang††

ABSTRACT

Recently, as application fields of software is extended, relative importance of software make a gradual increase and importance of development cost is being increased. However, as former evaluation model of development cost evaluate at the functional point of view for the most part, at this paper, I intend to propose evaluation model of software development cost COSMOS-Q(COSt MOdel for Subcontract-Quality) which one can evaluate also in quality as well as function. The model proposed in this paper set the goal at software orderer evaluate software cost exactly with only order specification information. At this paper, I proposed cost evaluation model and evaluated it's validity refering to review result in ISO/SC7 about a software quality feature with extraction of quality feature factor which produce change of cost and set up the evaluation measure adoptable as order condition.

1. 서 론

최근 소프트웨어가 차지하는 비중이 증가함에 따라 소프트웨어 개발 비용에 대한 중요성이 크게 인식되고 있다. 그러나 지금까지의 소프트웨어 개발 비용 견적은 우선 납입품의 원시 코드 규모 DSL(Delivered Source Lines of code)을 경험적으로 추정한 후에 소요 공수를 추정하여 산정하였다. 바꾸어 말하면, 규모로 환산 가능한 기능특성에 착안한 평가가 이루어져 왔기 때문에 보수성이나 이식성의 확보에 관한 품질 향상 노력의 평가가 불충분하였다. 본 연구에서는 이 점에 주목하여 소프트웨어의 품질특성에 관한 ISO/9126에서의 검토결과를 참고로 하여 비용의 변동요인이 되는 품질특성(비용요인)의 설정과 발주조건으로 규정가능한 평가척도의 설정을 시험하였다.

종래의 대표적인 평가척도로 규모와 품질면을

· 이 논문은 1993년도 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었음
† 종신회원 : 서울산업대학교 전자계산학과 강사
†† 종신회원 : 강원대학교 전자계산학과 교수
논문접수 : 1994년 2월 21일, 심사완료 : 1994년 5월 2일

고려한 Boehm의 COCOMO (COnstructive COst MOdel)가 널리 알려져 있다 〈표 1〉. 그러나 COCOMO는 개발 프로젝트의 운영방법과 요원의 기능을 수주자측이 아니면 파악하기 어려운 비용 요인이 있기 때문에 발주자측이 이것을 정확하게 파악하여 평가해야 하는 문제점을 가지고 있다. 또한 종래의 모델은 대부분이 소프트웨어의 가치를 기능면만을 고려한 것이었기 때문에 본 연구에서는 첫째, 소프트웨어의 가치를 기능면뿐만 아니라 품질면의 노력도 고려한 평가가 가능하고 둘째, 발주자가 발주 명세에 규정한 항목만을 이용하여 평가가 가능하도록 하며, 발주시의 비용 견적과 완료시의 생산성 평가를 할 수 있도록 소프트웨어 개발 비용의 평가 모델 COSMOS-Q (COSt MOdel for Subcontract-Quality)를 제안한다.

제안된 평가 모델 COSMOS-Q에 의해 다음과 같은 효과를 기대할 수 있다. 발주자는 발주명세에 규정된 비용요인 값을 이용하여 소프트웨어의 가치를 보다 객관적이고 정량적으로 평가함으로써 품질을 고려한 적정한 가격의 소프트웨어 발주가 가능하고 수주자측에서도 품질 향상을 위해 노력함으로써 보다 많은 이익을 제고할 수 있다. 본 연구의 구성은 제 2장에서 비용평가 모델 COSMOS-Q를 제안하고 제 3장에서는 소프트웨어의 비용을 산정하는데 사용되는 스텝단가에 영향을 미치는 단가요인의 추출과 평가척도를 설정하였고 그 타당성 평가를 하였다. 마지막으로 제 4장에서는 결론 및 향후 연구방향을 제시하였다.

2. 비용 평가 모델의 제안

2.1 메타 모델

일반적으로 소프트웨어의 비용은 개발하는 제품 양과 단위량의 공수에 의해 평가된다. 그러나 외주 개발의 경우, 발주자측에서는 공수를 정확하게 파악할 수 없기 때문에 다음 식과 같이 단위량당 비용을 이용하여 평가하는 것이 오히려 정확한 평가가 된다.

$$Y = M \cdot y \quad \cdots\cdots (1)$$

Y : 제품 비용, M : 제품 양, y : 단위량당 비용

〈표 1〉 COCOMO의 중간 모델
〈Table 1〉 Intermediate COCOMO

속 성	비 용 요 인	노력계수
소프트웨어 제품	RELY : Required software reliability; 소프트웨어에 요구되는 신뢰성	f_1
	DATA : Database size; 처리해야 할 데이타베이스의 사이즈	f_2
	CPLX : Product complexity; 소프트웨어 제품의 복잡성	f_3
컴 퓨 터	TIME : Execution time constraint; 시스템에 의한 실행시간의 제약	f_4
	STOR : Main storage constriant; 주기억의 제약(프로그램 메모리 평균점유율)	f_5
	VIRT : Virtual machine volatility; OS, 하드웨어의 변경 빈도	f_6
	TURN : Computer turnaround time; 프로그램 개발시의 컴퓨터 반환시간	f_7
요 원	ACAP : Analyst capability; 분석가(시스템 설계자)의 능력	f_8
	AEXP : Application experience; 동일한 어플리케이션의 경험 정도	f_9
	PCAP : Programmer capability; 프로그래머의 능력(설계후의 담당자)	f_{10}
	VEXP : Virtual machine experience; OS, 하드웨어의 경험 정도	f_{11}
	LEXP : Programming language experience; 사용된 프로그래밍 언어의 사용 정도	f_{12}
프 로 젝 트	MODP : Use of modern programming practices 소프트웨어 개발 기법의 사용 정도	f_{13}
	TOOL : Use of software tools; 사용 소프트웨어 툴의 내용	f_{14}
	SCED : Required development schedule; 요구되는 개발 기간의 제약	f_{15}

$EMM = C \cdot (KDSI)^b \cdot f_1 \cdot f_2 \cdot f_3 \cdots f_{15}$
EMM : 견적 공수(월간 필요 인원)
C : 개발 모드에 의해 결정되는 계수
KDSI : 납입 원시 코드 규모(K step)
f_i : 비용요인의 영향도를 나타내는 노력계수

2.2 제품 양(M)의 평가척도

개발 완료된 소프트웨어 제품은 보통 프로그램과 문서로 구성되므로 소프트웨어 제품의 양은 이들을 합산하여 나타낼 수 있다고 생각한다. 그러나 현재는 프로그램의 양과 문서의 양을 동일 척도로 취급하는 환산 방법이 없기 때문에 본 모델에서는 종래 납입품의 원시 코드 규모(DSL : Delivered Source Lines of code)를 제품의 양으로 사용한다.

$$M = DSL \quad \cdots\cdots (2)$$

M : 제품의 양, DSL : 원시 코드 규모

그러나 문서는 프로그램의 품질에 크게 영향을 주기 때문에 본 모델에서는 단가요인의 평가척도에 이용한다.

DSL은 〈표 2〉와 같은 코드를 그 측정 후보로

고려할 수 있다. 여기서 주석문, 공백문 등은 문서와 마찬가지로 단가요인의 평가척도로 이용하는 것이 적당하기 때문에 DSL 대상에서 제외한다. DSL 추정은 기능 요소로부터 추정하는 방법을 이용하였다 [2, 3, 4, 5]. 그러나 DSL은 3장에서 기술하는 규모 요인에 의존하여 그 값이 결정되기 때문에 이들을 고려한 추정방법에 대해 검토가 필요하다.

〈표 2〉 DSL의 측정 후보 코드
〈Table 2〉 Measure Candidate Code of DSL

내역				측정 대상코드
I	II	III	IV	
납품프로그램	프로그램 본체	소스 (source)	data 선언문	○
			실행문	
			macro 문	
			compiler 문	
			debug 문	×
			주석문	×
			공백문	
		JCL		○
	데이타			○
	중간 생산물	테스트 프로그램	단위 시험용	×
			결합 시험용	
		테스트 데이타	단위 시험용	
			결합 시험용	
		툴(Tool)	드라이버	×
			기타 툴	

(주) ○ : 측정대상, × : 측정 대상외(발주시 지정경우 제외)

2.3 단위량당 비용(y)의 평가척도

제품 양의 척도를 DSL로 했을때 단위량당 비용은 프로그램의 스텝당 비용 즉, 스텝단가가 된다. 본 연구에서의 스텝단가는 3장에서 기술한 여러 품질향상 노력의 정도에 따라 그 값이 변동하는 것으로 간주한다. 품질향상 노력중 측정가능한(발주시에 노력의 레벨을 지정가능) 요인을 단가요인(Q_i)이라 하고, 단가요인(Q_i)별로 평가척도를 정하여 그 값에 따라 정해지는 단가에의 영향도(노력계수)를 q_i라고 하면, 스텝단가는 다음 식으로 표현할 수 있다.

$$y = K \cdot \prod_{i} q_i \quad \cdots\cdots (3)$$

K : 정수(프로그램의 종별에 따라 정해지는 표준적인 스텝단가)

q_i : 단가요인별로 평가척도의 값에 따라 정해질 수 있는 노력계수

i : 노력계수의 수

2.4 COSMOS-Q 평가식

이상으로부터 소프트웨어 제품의 비용(Y)은 다음 식으로 나타낸다.

$$Y = K \cdot DSL \cdot \prod_{i} q_i \quad \cdots\cdots (4)$$

3. 단가요인의 추출과 평가 척도의 설정

3.1 단가요인의 추출

스텝단가에 영향을 미치는 단가요인(Q_i)을 추출하기 위해 ISO/IEC JTC1/SC7에서 검토한 품질특성 항목을 참고로 하여 다음 두가지의 비용요인으로 대별한다.

(1) 규모요인 … 프로그램의 규모에 영향을 미치는 요인

(2) 단가요인 … 품질향상 노력을 나타내는 요인

품질특성 항목중 14개의 특성 항목을 규모요인으로 분류하고 나머지 12개의 특성 항목을 단가요인 후보로 추출하였다. 단가요인 후보 중 측정가능한 추적 가능성, 일관성, 자기 기술성 등 9개의 특성을 단가요인(Q_i)으로 추출하여 그 결과를 〈표 3〉에 나타내었다.

특히, 각 품질특성 항목이 어느 요인에 속하는가의 판단은 다음과 같이 하였다. 즉, 기능적으로 같은 소프트웨어일지라도 기능 추가와 오류 추적의 용이성(추적 가능성), 문서 판독의 용이성(일관성), 보수 용이성(모듈성) 등, 납품 후 발주자가 유지보수시 효과적인 제품 성질을 가지는 측정 요인을 단가요인으로 하였고, 계산 정확성과 접근 가능성, 견고성 등 발주명세서에 실현기능을 지정하여 프로그램 규모에 반영되는 성질을 가지는 요인을 규모요인으로 하였다.

3.2 단가요인의 평가척도

3.1절에서 추출된 9개의 단가요인에 대한 평가척도와 노력계수 q_i 범위의 설정방법을 다음과 같이 제안한다. 이 평가척도의 측정을 기본으로

<표 3> ISO의 소프트웨어 품질특성 분류에 기초한 비용요인의 추출

<Table 3> Extract of Cost Factor Based on Software Quality Characteristics Classification of ISO

품질특성			내부특성의 개요	비용요인의 분류		비고
외부특성	관계	내부특성		규모요인	단가요인	
기능성		완전성	요구사양서에 기술된 기능이 빠짐없이 실현되어 있는가를 나타내는 성질	○		
		추적 가능성	요구 명세서로부터 각종 설계서와 프로그램 및 설명서에의 기능의 관계가 추적되는 성질		◎	Q_1
		일관성	소프트웨어를 개발·설계할 때의 설계방법, 코딩방법, 문서 기술방법의 일관성있는 성질		◎	Q_2
		자기 기술성	소프트웨어가 가지고 있는 기능 및 기능간의 관계를 프로그램 자신과 설명서, 문서가 설명하고 있는 성질		◎	Q_3
신뢰성		무예속성	시스템과 프로그램이 갖고 있는 동질목적의 기능이 모순을 포함하고 있지 않는 성질	○		
		계산 정확성	계산결과, 출력이 요구정도의 달성 성질	○		
		데이타 공용성, 통신절차공용성	시스템내/시스템간에서 사용하는 데이타, 통신순서 인터페이스가 공용되는 성질	○		
		접근 가능성	프로그램 성능과 관련장치를 선택하여 자유롭게 사용할 수 있는 성질	○		
		접근 제어성	Access ID와 Command를 체크해 소프트웨어와 데이타에서 접근을 제어할 수 있는 성질	○		
보수성		접근 감사성, 계측성	소프트웨어와 데이타의 접근기록을 남겨서 프로그램의 동작상황을 판찰할 수 있는 성질	○		
		견고성, 적합성, 안정성	조작 오류와 내부이상의 발생에 대해 데이타와 프로그램이 파괴되지 않는 성질	○		
		모듈성, 구조성	소프트웨어가 적절하게 구조화되어 있고, 프로그램의 변경이 국소적으로 되는 성질		◎	Q_4
이식성		단순성	기능을 쉽게 이해할 수 있는 성질	○		
		자기 포함성	다른 프로그램에 의존하지 않고 기능을 수행 가능하게 하는 성질	○		
		확장성	사양의 추가·변경에 대해 용이하게 대응할 수 있도록 준비되어 있는 성질	○		
		환경 독립성	프로그램이 특정한 소프트웨어, 하드웨어, 데이타의 환경에 의존하지 않고 동작가능한 성질		◎	Q_5
		제품 관리성	제품을 바르게 관리할 수 있는 성질	○		
		통일성 전달성	의미, 표현, 순서가 통일적, 동질적인 성질 입출력 형식과 내용이 통일되는 성질	○		
사용성		표현성, 계층성, 비유성, 주목성	입출력과 처리논리, 결과를 잘 표현하는 성질		○	
		적시성, 적당성	사용자와의 시간적, 양적관계를 양호하게 하는 성질		○	
		명료성, 간단성	불필요한 정보없이 정보가 콤팩트한 성질		○	
		완비성	소프트웨어의 사용을 돕기 위한 기능과 사물이 갖추어져 있는 성질		◎	Q_6
		설명성, 선택성, 유도성, 생략성, 환경 적합성	필요한 정보를 제공하여 사용자의 능력이나 기호에 따라 소프트웨어가 갖는 논리에 사용자를 유도해 투입노력이 적게 커스토마이즈가 용이한 성질	○		
효율성		처리 속도	처리결과를 얻기까지의 처리속도의 성질		◎	Q_7
		처리 능력	단위시간에 처리할 수 있는 작업량의 성질		◎	Q_8
		자원 사용량	처리에 필요한 하드웨어 양의 성질		◎	Q_9

(주) ◎ : 단가요인 항목, ○ : 규모요인 항목 또는 측정 불가능한 단가요인 항목

q_i를 정하고, 2장에서 제안한 모델식에 의해 발주시 견적 및 납품시 평가를 정량적으로 하는 것이 가능하다.

(1) 추적 가능성(Q_1)

각 개발공정에서 명세화된 기능이 다음 공정에 정확히 전개되었는지의 정도를 나타낸 것이다. 설계단계에서의 불충분하고 부정확한 명세가 다음 공정으로 넘겨지면 그 복구에 많은 공수가 걸리게 된다. 따라서 설계단계의 추적가능성을 중시하여 '스텝당 설계 문서의 리뷰 지적항목수'를 평가척도로 채용한다.

- ○ 평가척도 1 : 리뷰 지적 항목수 / DSL
- ○ 노력계수 q_1 범위 : 척도의 값이 작을수록 품질이 좋고 q_1이 높다.

(단, 상세한 리뷰를 실시한다는 전제)

(2)일관성(Q_2)

소프트웨어의 개발방법, 절차, 문서화 등에 관한 일관성의 정도를 나타낸 것이다. 개발방법, 절차 등을 규정한 작업표준이 없으면 개발기술이 불안정해지고 기능의 누락, 준비의 부족, 부적합 등이 발생하기 쉽다. 이러한 점으로부터 일관성의 평가 척도로 '개발 작업 표준의 적용도'를 채용한다.

- ○ 평가척도 2 : 개발작업 표준의 적용도
- ○ 노력계수 q_2 범위 :

· 품질이 좋다 ↑	· 전공정 적용(발주측 표준)
q_2가 높다	· 부분 적용(발주측 표준)
	· 전공정 적용(수주측 표준)
· q_2가 낮다	· 부분 적용(수주측 표준)
품질이 나쁘다 ↓	· 적용 안함

(3)자기 기술성(Q_3)

생산품에 소프트웨어의 기능이나 기능간 관계의 이해를 돕는 내용의 기술이 어느정도 되어있는가의 정도를 나타낸 것이다. 즉, 자기 기술성이 부족되면 소프트웨어의 변경이나 개조가 어렵게 된다. 일반적으로 이 기술들은 설계 문서와 원시 프로그램중에 기술되어 있기 때문에, 자기 기술성의 평가 척도로서 '스텝당 설계 문서 매수'와 '스텝당 원시 코드중의 주석 코드 행수'를 채용한다.

- ○ 평가척도 3(1) : 설계 문서 매수/DSL
- ○ 평가척도 3(2) : 주석 코드 행수/DSL
- ○ 노력계수 $q_{3(1)}$, $q_{3(2)}$범위 : 척도의 결과는 클수록 품질이 좋고 $q_{3(1)}$, $q_{3(2)}$도 높다.

(4) 모듈성(Q4)

모듈성은 프로그램의 이해를 용이하게 하고, 보수성과 이식성을 확보하기 위한 모듈 설계의 적절성의 정도를 나타내기 위한 것이다. 모듈성의 평가척도로 모듈 분할의 적절성을 나타내는 '모듈 응집도', '모듈 결합도', '평균 모듈 사이즈'를 채용한다.

- ○ 평가척도 4(1) : 모듈 응집도(모듈 내 구성요소의 결합 강도를 나타낸다)
- ○ 노력계수 $q_{4(1)}$ 범위 : 강도가 강할수록 품질이 좋고 $q_{4(1)}$이 높다.

(프로그램을 구성하는 모듈중에서 가장 응집도가 약한 모듈을 프로그램 전체의 응집도로 한다)

프로그램의 품질을 측정하는 과정에서 어떤 모듈의 응집도를 평가하게 되면 그 모듈의 품질수준이 결정되며, 이 때 품질을 개선하기 위해 모듈이 좀 더 강한 응집도를 갖도록 변환할 필요가 있다. 변환 방법은 모듈이 어떤 응집도를 가지고 있는가에 따라 결정되며 모듈내의 응집도의 정도와 대처 방법은 다음과 같다.

평 가	모듈 응집도	대처 방법
· 품질이 좋다	· 기능적 응집도	· 그대로 둔다
$q_{4(1)}$가 높다	· 순차적 응집도	· 분할하지 않아도 좋다
↑	· 통신적 응집도	
	· 절차적 응집도	· 분할하는 쪽이 좋다
↓	· 시간적 응집도	
· $q_{4(1)}$가 낮다	· 논리적 응집도	· 분할해야 한다
품질이 나쁘다	· 우연적 응집도	

- ○ 평가척도 4(2) : 모듈 결합도(모듈간 결합정도의 느슨함을 나타낸다)
- ○ 노력계수 $q_{4(2)}$ 범위 : 결합도가 약할수록 품

질이 좋고 $q_{4(2)}$가 높다.

(프로그램 중에서 가장 결합도가 강한 정보적 결합을 해당 프로그램의 결합도로 한다)

프로그램의 품질을 측정하는 과정에서 모듈간의 결합도를 평가하게 되면 해당 모듈간의 품질 수준이 결정되며, 이 때 품질을 개선하기 위해 모듈이 좀 더 약한 결합도를 갖도록 변환할 필요가 있다. 변환 방법은 모듈이 어떤 결합도를 가지고 있는가에 따라 결정되며 모듈간 결합의 정도와 대처 방법은 다음과 같다.

평 가	모듈 결합도	대처 방법
· 품질이 좋다 $q_{4(2)}$가 높다	· 자료 결합도 · 구조 결합도	· 그대로 둔다
	· 제어 결합도 · 공통 결합도	· 일단 통합하여 재분할을 검토
· $q_{4(2)}$가 낮다 품질이 나쁘다	· 내용 결합도	· 1개 모듈로 통합한다

○ 평가척도 4(3) : 모듈 사이즈

○ 노력계수 $q_{4(3)}$ 범위 : 척도의 값이 작을수록 품질이 좋고 $q_{4(3)}$이 높다.

(5) 환경 독립성(Q_5)

프로그램의 환경(하드웨어 기종, OS, 언어 등)의 의존도를 정량화하는 것으로 값이 작을수록 이식이 용이해지고 품질이 높아진다. 이 의존도를 측정하는 척도로 OS와 언어의 표준 사양과의 일치정도로 평가하는 방법이 있으나, 발주시에 정량적인 지정이 어려운 문제점을 가지고 있다. 따라서 환경 독립성의 평가척도로 '스텝당 환경 의존 원시 코드량'을 채용한다.

○ 평가척도 5 : 환경의존 DSL/DSL

○ 노력 계수 q_5의 범위 : 척도의 값이 작을수록 품질이 좋고 q_5가 높다.

(6) 완비성(Q_6)

소프트웨어를 이용하는 사용자를 도와주는 수단의 충실도를 나타내는 것으로 매뉴얼의 충실도로 나타내도록 한다. 따라서 완비성의 평가척도로 '스텝당 매뉴얼 매수'를 채용한다.

○ 평가척도 6 : 매뉴얼 매수/DSL

○ 노력계수 q_6 범위 : 척도의 값이 클수록 품질이 좋고 q_6이 높다.

(단, 사용자 매뉴얼의 품질 확보를 위한 기술 가이드라인 등에 따를것)

(7) 처리속도(Q_7)

소프트웨어의 실행범위를 측정하는 척도의 하나로 응답시간이 있다. 이것은 소정의 처리를 실행하는 빠르기로서 프로그램의 구성, 처리 알고리즘 등에 의해 좌우되는 반면, 동작 환경인 하드웨어와 동시 이용수 등의 이용특성에서도 영향을 받는다. 따라서 처리속도의 평가척도로 절대값으로서의 응답시간이 아닌 요구조건으로서의 허용 응답시간에 대한 단축정도로 계산한다.

○ 평가척도 7 : 실현 응답시간/허용 응답시간

○ 노력계수 q_7 범위 : 척도의 값이 작을수록 품질이 좋고 q_7이 높다.

(8) 처리능력(Q_8)

처리능력(throughput)은 단위시간에 수행할 수 있는 작업량을 측정하는 척도를 나타내는 것이다. 이 처리능력은 처리속도와 같이 동작 환경 등의 영향을 받기 때문에 절대값으로서의 처리능력은 평가척도가 되기 어렵다. 따라서 요구조건으로서의 허용 처리능력에 대한 실현 가능한 처리능력으로 계산한다.

○ 평가척도 8 : 실현 문서/허용 문서

○ 노력계수 q_8 범위 : 척도의 값이 클수록 품질이 좋고 q_8이 높다.

(9) 자원 사용량(Q_9)

소프트웨어가 목적으로 하는 처리의 실행과 자원 사용 효율의 양호함을 나타내는 것이다. 자원 사용량은 적은 편이 효율적이나 소프트웨어의 종별 등에 따라 그 값을 적절히 규정하는 것이 바람직하다. 따라서 요구조건으로서의 자원 사용량에 대한 절약도로서 계산한다. 주기억 사용량과 CPU 자원 사용량의 예를 들면 다음과 같다.

○ 평가척도 9(1) : $\frac{\text{실제 최대 주기억 사용량}}{\text{허용 주기억 사용량}}$

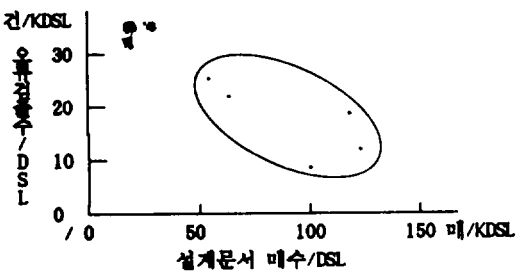
○ 평가척도 9(2) : $\frac{\text{실제 최대 CPU 사용량}}{\text{허용 CPU 사용량}}$

○ 노력계수 $q_{9(1)}$, $q_{9(2)}$ 범위 : 척도의 값이 작을수록 품질이 좋고 $q_{9(1)}$, $q_{9(2)}$가 높다.

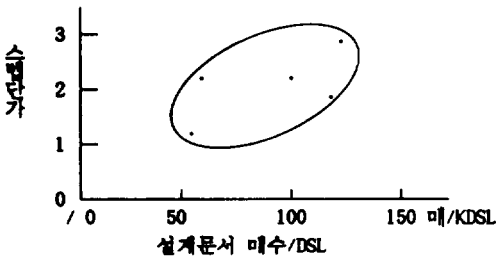
3.3 척도의 타당성 평가

자기 기술성(Q_3)을 예로 3.2절에서 설정했던 평가척도의 타당성을 유사한 5개의 프로그램을 예로 그 관계를 살펴보았다. 자기 기술성의 평가척도에서는 단위 규모당 설계 문서수가 많을수록 품질이 좋고 스텝단가는 높다고 가정하고 있다.

(그림 1)은 샘플 케이스에 대해서 설계 문서 매수와 오류 검출수의 관계를 나타낸 것으로 단위규모당 설계문서수가 많을수록 오류 검출수가 적어져 품질이 높아지는 경향이 나타나 있다. 또 (그림 2)는 설계 문서 매수와 스텝단가의 관계를 본 것으로 설계문서수가 많아질수록 스텝단가는



(그림 1) 설계 문서 매수와 오류 검출수와의 관계
(Fig. 1) Relation between the number of Design Document and Error Detection



(그림 2) 설계 문서 매수와 스텝단가와의 관계
(Fig. 2) Relation between the number of Design Document and Step Unit Cost

높아지는 경향을 알 수 있다.

그러나 샘플수를 증가시켜 이 경향의 확실성을 확인할 필요가 있으며, 다른 평가척도 설정 방법과의 비교 검토도 필요하다고 본다.

4. 결 언

기존의 비용 평가 모델의 대부분이 소프트웨어 수주자측 입장에서 제안된 것에 비해, 본 연구에서는 발주자측 입장에서 모델화의 가능성을 추구하고자 하였다. 또한, COSMOS-Q에서 채택한 비용요인은 발주시 명확하게 규정하는 것이 바람직하나 단가요인에 대해서는 규정하기 어려운 경우도 있다고 생각된다. 따라서 본 모델 도입을 기회로 각 요인에 대해서 발주조건이 명확히 규정되어지므로 원하는 품질을 갖춘 소프트웨어의 입수가 가능함과 동시에 모델을 커스토마이즈하기 위한 귀중한 자료를 얻을 수 있다.

앞으로의 연구 과제로는, 본 모델을 확립하기 위해 필드 데이타의 측정과 분석수단의 정비, 필드 데이타의 분석에 기초한 단가요인의 노력계수 q_i의 범위 설정, 규모요인 분석에 의한 DSL의 추정 정도의 향상 등이 필요하다고 본다.

참 고 문 헌

[1] B. W. Boehm, 'Software Engineering Economics', Prentice-Hall, Inc, 1981.

[2] B. W. Boehm, P. N. Papaccio, "Understanding and Controlling Software Costs", IEEE Trans. on Software Eng., Vol. 14, No. 10, 1988.

[3] L. H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimation Problem", IEEE Trans. on Software Eng., Vol. 4, pp. 345-361, 1978.

[4] A. Albrecht and J. Gaffeny, "Software Function Source Lines of Code, and Development Effort Prediction, A Software

Science Validation", IEEE Trans. on Software Eng., Vol. 9, No. 6, 1983.

[5] M. Itakura and A. Takayanagi, "A Model for Estimating Program Size and its Evaluation", Proceedings, 6th international Conference on Software Engineering, pp. 104-109, Sep, 1982.

[6] D. B. Brown, "A Cost Model for Determining the Optimal Number of Software Test Cases", IEEE Trans. on Software Eng., Vol. 17, No. 2, 1989.

[7] C. Jones, "Programming Productivity", Issues for the Eighties, IEEE Compu-ter Society Press, 1981.

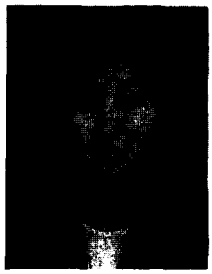
[8] B. Zimmer, "Software Quality and Productivity Analysis at Hwelett Packard", COMPSAC89, 1989.

[9] V. Basili and M. Zelkowitz, "Analyzing Medium Scale Software Development", Proceedings of the 3rd International conference on Software Engineering", IEEE, pp. 116-123, 1978.

[10] G. J. Myers, "Reliable Software through composite design", petrocell:/Charter, 1975.

[11] 양해술 외, "객체지향 모듈의 품질평가에 관한 연구", 과기처 특정연구비 지원 최종 보고서, 1991.

[12] 양해술 외, "소프트웨어 품질 평가체계와 자동화 도구의 개발", 한국통신 '94장기기초 연구과제(협약서 94-1) 중간보고서, 1994.



이 용 근

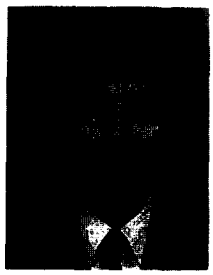
1988년 강원대학교 자연과학대학 전자계산학과 졸업(이학사)

1994년 강원대학교 전자계산학과 소프트웨어공학 전공(이학석사)

1989년-1992년 강원대학교 전자계산학과 조교

1994년~현재 한림전문대학, 서울산업대학교 전산계산학과 강사

관심분야 : 소프트웨어공학(특히, 소프트웨어 품질보증과 품질평가, 객체지향 프로그래밍, 객체지향 분석과 설계)



양 해 술

1975년 홍익대학교 공과대학 전기공학과 졸업(학사)

1978년 성균관대학교 정보처리학과 정보처리 전공(석사)

1991년 日本 오사카대학 정보공학과 소프트웨어공학 전공(공학박사)

1975년~79년 육군중앙경리단 전자계산실 근무

1984년~92년 성균관대학교 경영대학원 강사

1986년~87년 日本 오사카대학 객원연구원

1994년~현재 한국산업표준원 이사

1994년~현재 한국정보과학회 학회지 편집부위원장

1994년~현재 한국정보처리응용학회 논문편집위원장

1980년~현재 강원대학교 전자계산학과 교수

관심분야 : 소프트웨어 공학(특히, S/W 품질보증과 평가, SA/SD, OOA/OOD/OOP, CASE), 소프트웨어 프로젝트관리.