# IT산업 생산성 향상을 위한 프로젝트 실행계획 수립 방안 연구 - COCOMO II 적용사례

박 철 구* · 김 창 은*
*명지대학교 산업경영공학과

# A study of actual planning how to increase IT productivity by COCOMO II Model

Cheol-Gu Park* · Chang-Eun Kim*
*Department of Industrial and Management Engineering, Myongji University

## Abstract

Project implementation plan is a blueprint that confirms project performance activities and specifies required man-hour, period and resource imput ratio. Various figures, the results of implementation plan, are predicted through estimation, and because of superiority of objectivity and repeatability, numerical formula-based estimation model is often used overseas.

COCOMO model is the representative estimation model whose theories and formulas are publicized and it predicts the total man-hour required for software system development. This model is publicized in "Software Engineering Economics" written by Professor Barry Boehm of the U.S., and is the most widely applied numerical formula-based estimation model. This study is conducted to provide a series of methods that are optimal for KTDS environment by choosing COCOMO II model among various types of COCOMO models. In establishing implementation plan, COCOMO II model alone is not sufficient, it is necessary to link with and apply standard WBS system and standard man-hour. In establishing specific implementation plan, phased standard WBS system in order of the first phase of all the activities implemented in the project, Activity, Task, and Role, and the man-hour put into this should be distributed according to standard ratio from COCOMO II model's total man-hour. This study provides explainations by establishing standard WBS system and linking with COCOMO II model.

**Keywords : 실행계획, COCOMO II, 표준공수, 생산성**

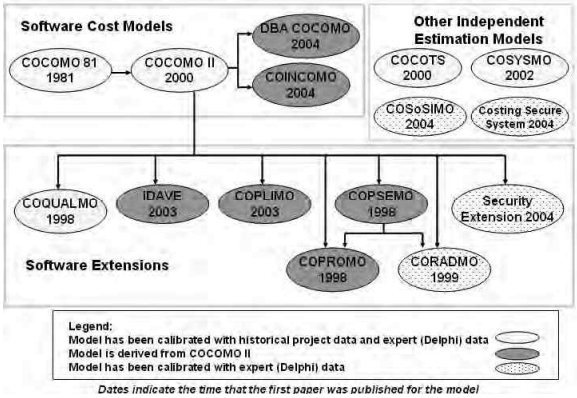
## 1. 서 론

IT기업의 생산성 측정을 위한 COCOMO 모델은 이론과 수식이 공개된 대표적인 견적모델로서 소프트웨어 시스템개발에 소요되는 총 공수를 예측 가능하게 한다.

이 모델은 미국의 Barry Boehm교수가 저서"Software Engineering Economics"를 통해 발표하였으며(1981), 현재 가장 광범위하게 적용되는 수식기반의 견적모델이다. 1981년 발표된 초기모델은 폭포수(Waterfall) 개발 모델 기반의 COCOMO I 모델이었고, 1987년에 발표된 모델은 Ada 언어 기반 프로젝트의 견적을 위한 Ada COCOMO 모델 이었다. 데이터를 기반으로 COCOMO I 모델을 개선한 COCOMO II 모델은 1997년에 처음 발표되었는데, 이 버전은 현재 계속 진화하고 있다.

실행계획을 수립하려면, COCOMO II 모델 단독으로는 부족하며, 표준WBS체계와 표준공수를 연계 적용하는 것이 필요하다. COCOMO II 모델은 총 공수를 예측하여 주고 이를 단순히 최상위 레벨 공정 단계로 분배하기 때문에, 실행계획이 될 수 없다. 구체적인 실행계획을 수립하려면 먼저 프로젝트에서 수행하는 모든 활동을 단계, Activity, Task, Role 순의 표준 WBS 체계가 필요하며, 이에 투입되는 공수가 COCOMO II 모델

† 교신저자: 박철구, 경기도 용인시 처인구 남동 산 38-2 명지대학교 제1공학관 537호
M · P: 010-7247-8474, E-mail: fervorman@kt.com
2010년 4월 20일 접수; 2010년 6월 1일 수정본 접수; 2010년 6월 3일 게재확정




[그림. 1] COCOMO Model 변화

의 총 공수로부터 표준비율에 따라 분배되어야 한다.

이에 대해 본 논문에서는 생산성 향상을 위한 COCOMO Ⅱ를 적용한 총 공수측정과 이에 표준WBS체계의 접목을 통하여 효과적인 프로젝트 관리를 통한 생산성 향상 모델을 개발하고자 한다.

## 2. COCOMO Model

### 2.1 COCOMO Model

Boehm이 1982년에 제안한 cost estimation모델로서, 소프트웨어비용 또는 개발자원투입량(development effort)을 평가하는 대표적인 비용평가모델로서 발전해왔다[그림. 1].

Boehm은 COCOMO Model을 basic, intermediate, advanced model 3가지로 구분하여 제안하였으며, 프로젝트의 특성에 따라서 organic(소형 프로젝트에 적합함), semi detached(중형 프로젝트에 적합하며, 데이터베이스 시스템을 사용함),embedded mode(대형 프로젝트에 적합하며, 자체 개발환경을 가지고 수행함.)로 클래스를 달리하고 15개의 “cost driver”를 사용하여 6단계의 범위(scale)로 EAF(Effort Adjustment Factor)를 두어 평가한다[5].

기본 모델식은 (1)~(3)과 같다.

$E = ab(KLOC)exp(bb)$ (1)

$D = cb(E)exp(db)$ (2)

$E = ai(KLOC)exp(bi)EAF$ (3)

E : effort person month

D : project duration

EAF : range(0.9~1.4)

### 2.2 COCOMOⅡ Model

COCOMO Ⅱ PA(Post Architecture) 모델은 개발된 소스 및 규모요인(scaling driver)과 가격요인(cost driver) 등을 이용하여 프로젝트의 생산성, 가격 등을 결정하는 모델이다. COCOMOⅡ 의 공수 수식은 다음과 같다.

$PM\ NS = A \times Size \times \prod_{i=i}^{n} EMi$

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

$TDEV\ ns = C \times (PMNS)F$

Where $F = D + 0.2 \times 0.01 \times \sum_{j=1}^{5} SFj$

$= D + 0.2 \times (E\ \ B)$

EMi : effort multipliers of cost driver

B : multipliers scaling driver

A : constant

즉, 소프트웨어 개발에 필요한 소요 공수(E: Effort in Man Months, 또는 MM으로 표기)의 예측 및 비용 산정을 목적으로 1981년 Dr. Barry Boehm에 의해 최초의 COCOMO가 제안 되었으며, 현재의 정보기술을 반영하여 1995년 수정된 모델인 COCOMO Ⅱ 모델을 발표하였다. COCOMOII 모델은 시스템 개발 주기 중 각 단계별로 다음과 같이 세 가지 모델을 적용할 수 있다.

**1) Application Composition 모델**

개발 초기 프로토타입과 시제품 개발 시 적용 가능한 모델

**2) Early Design 모델**

개발할 제품의 규모를 알기 어려운 프로젝트의 초기 단계에 기능점수(Function Point)와 같은 사이징 메트릭(Sizing Metrics)을 이용하는 모델

**3) PA(Post Architecture) 모델**

프로젝트 개발이 완료되어 소프트웨어 규모 및 가격요인들을 정확하게 적용할 수 있는 모델 또 프로젝트의 소프트웨어 개발 규모를 알고 있는 경우는 PA 모델 적용이 가능하며, PA 모델에서 고려할 변수들은 다음과 같다.

(1) Total source line 수(SLOC)

(2) 규모요인(Scale Drivers)

Precedent, Flexibility, Process Maturity, Architecture /Risk Resolution, Team Cohesion

(3) 가격요인(Cost Drivers)

① Product Factors: Required Software Reliability, Database Size, Product Complexity, Required Reusability, Documentation Match to Life cycle Needs

② Platform Factors: Execution Time Constraint, Main Storage Constraint, Platform Volatility

③ Personnel Factors: Analyst Capability, Programmer Capability, Application Experience, Programmer Experience, Language and Tool Experience, Personal Continuity

④ Project Factors: Use of Software Tools, Multi state Development, Required Development Schedule

그 외에 인건비산출은 아래의 식으로 표현된다.

직접인건비 = 소요공수(MM)×기술자의 월급여

위에서 소개한 변수들 외에 요구 사항의 변경으로 일부 코딩 되었던 부분을 수정해야 하는 경우 Breakage 변수를 고려할 수 있으며, 기존의 소프트웨어를 수정해서 재사용할 경우 기존 소프트웨어 수정 비율(percent design modified, percent code modified, percent of integration required), 소프트웨어의 이해 정도(software understanding) 등의 변수들이 고려된다.

## 2.3 생산성 모델에 따른 Cost Drivers

COCOMOII 모델에서는 기본적으로 17가지 기본 Cost Drivers를 운영하고 있으며, 이를 개발과 유지보수로 나누어 적용한다. 이에 KTDS는 COCOMOII에서 정의한 Architecture Model을 유지보수분야에 적용하고, Early Design 모델을 개발에 적용하여, Post의 각 도메인마다 생산성 및 관련 자료를 도출하였다.

### 1) Post Architecture Model

이 모델은 설계단계 이전 모델(early design model)보다 좀 더 정교한 실제적인 소프트웨어개발 노력도와 나아가서는 유지보수노력도 까지 측정할 수 있는 모델이다.

이 모델은 이미 소프트웨어 개발 생명주기의 구조구현 단계가 종료되었을 경우에 적용되면, 개발 비용을 산정하는데 매우 효과적이다. 노력승수(EM, Effort Multiplier)의 17가지 원가요소(cost drivers)와, 비례요인(SF, Scale Factor)을 결정하는 5가지 원가요소를 제시하고 있다.

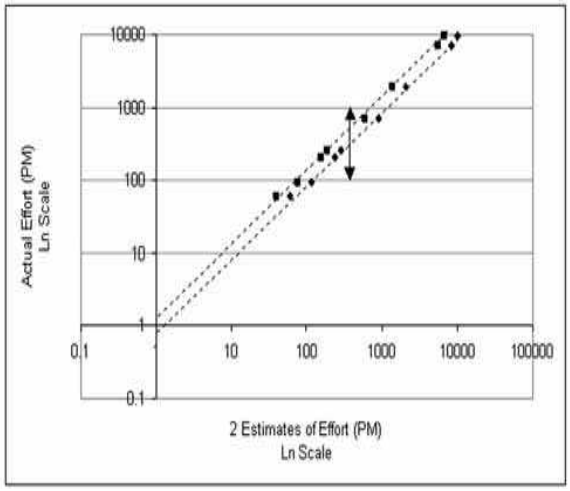
### 2) Early - Design Model

17가지 원가요소(cost drivers)들을 소프트웨어 제품관련 요소, 플랫폼 성능 제약 관련 요소, 투입 인력 관련 요소, 프로젝트 관리 관련 요소 등으로 분류할 수 있다.

[표. 1] 선형 입력인자간 관계

| Driver Group | Early Design | Post Architecture |
|---|---|---|
| Product | RCPX | RELY |
| | | DATA |
| | | CPLX |
| | | DOCU |
| | RUSE | |
| Platform | PDIF | TIME |
| | | STOR |
| | | PVOL |
| Personal | PREX | ACAP |
| | | PCAP |
| | | PCON |
| | PERS | APEX |
| | | PLEX |
| | | LTEX |
| | FCIL | TOOL |
| | | SITE |
| | SCED | SCED |

주) Driver 설명은 Appendix 참조



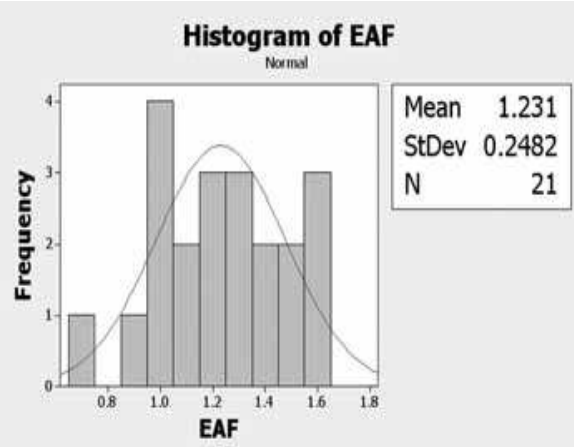

[그림. 2] 보정계수 함수의 기울기(Slope)

## 3. COCOMO II 신뢰성

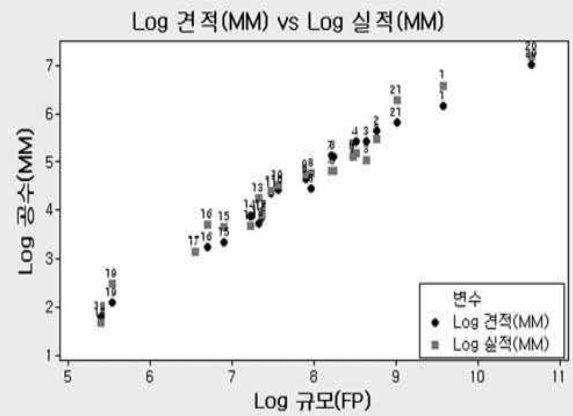
공수견적 수식의 보정은 [수식 1]에서 상수 A를 조정함으로써 가능해진다 [1]. 상수 A의 기본값은 2.94인데, 이 수치를 KTDS 고유의 개발환경 특성이 반영될 수 있도록 조정해야 한다. 상수 A를 조정한다는 것은 [그림 2]와 같이 견적과 실적 데이터 간 함수의 기울기(Slope)를 위·아래로 조절하는 것과 같다.

경우에 따라 COCOMO II 모델의 입력인자 자체를 특정 개발조직 환경에 맞추어 변경할 수 있다[1]. 조사된 데이터에서 개발역량 정도를 표현하는 선형 입력인자인 PERS와 개발경험의 정도를 표현하는 성형 입력인자인 PREX는 차이를 특별히 인정할 수 없어서 두 인자를 통합하였다. 또한, 저장장치의 제한, CPU 사용시간의 배분, 플랫폼의 변경 등의 정도를 나타내는 선형입력인자인 PDIF는 국내 환경에서는 큰 영향이 없는 것으로 판단하여 입력인자 대상에서 제외하였다. 보정된 모델의 정확도 검증을 위해 모델 입력인자에 실적값을 적용해야 하는데, 이를 위해 모델 입력인자와 <표 1>의 생산성 영향인자를 대응시켰다. 대응관계는 선형모델 입력인자들과의 유사성을 대비하여 결정하였다.

보정된 COCOMO II 모델의 선형 입력인자(EMi)는 조사 데이터와의 대응관계를 적용하여 측정된 생산성 영향인자를 입력하였으며, 모델의 비선형 입력인자(SFj)는 기준 값을 적용하여 입력하였다. 규모는 측정된 기능점수 값을 적용하였는데, 모두 객체지향(개발)언어(Object Oriented Language)의 변환 값을 적용하여 도출된 소스코드 라인수(Line of code, LOC)를 입력하였다. 하나의 견적에서 입력되는 모든 선형 입력인자의 곱은(ΠEMi)은 총보정영향도인데 본 논문의 실적 데이터는 [그림 3]과 같은 결과를 나타낸다. 총 영향도의 평균이 아무런 영향을 주지 않는 기본값 '1'보다 큰 '1.23'이기 때문에 데이터 샘플이 보편적으로 난이도가 높은 프로젝트 위주임을 알 수 있다[2,5].




[그림. 3] 선형 입력인자(EMi)의 곱




[그림. 4] 견적(MM) vs 실적(MM)

보정한 모델수식을 적용하여 도출한 견적 대비 실적 값의 비교결과는 [그림 4]와 같다.

Boehm이 제시한 MMRE(Mean Magnitude of Relative Error)은 절대오차의 평균값이며, PRED(30%) (Prediction Level at 30%)는 절대오차가 30%미만인 항목의 비율이다[5].

여기서 시사하는 것은 시험 프로젝트로 나타난 절대오차 평균이 26%이며, 절대오차가 30% 이하인 프로젝트의 비율이 전체에서 62%라는 점이다. 1997년에 발표된 COCOMO Ⅱ 최초 버전의 PRED(30%)가 52%였던 것을 고려하면[3], 이번 결과는 COCOMO Ⅱ 모델의 한국화가 충분히 가능하다는 것을 보여준 것으로 생각된다. 향후, 모델의 정확도를 높이기 위해 추가 데이터의 확보 및 보정이 필요하며 이는 MMRE가 30%미만, PRED(30%)가 70% 이상 되어야 보편적으로 정확한 견적모델로 인식되기 때문이다[1].

## 4. 생산성 지표 설계

위와 같이 Boehm의 COCOMOⅡ모델을 국내 프로젝트의 생산성 모델의 신뢰성을 근거로 KTDS의 생산성 계산에 대한 Cost Driver는 [표2]와 같이 설정하였다.

[표. 2]. KTDS Early Design에 의한 Cost Driver

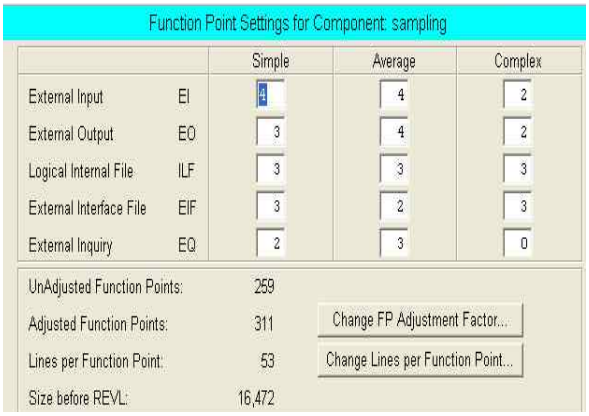
| DRIVER GROUP | Driver | '09년 | '10년 | '11년 |
|---|---|---|---|---|
| Personal | PERS | VERY HIGH | EXTRAHIGH | EXTRAHIGH |
| | PREX | HIGH | HIGH | VERY HIGH |
| Platform | PDIF | NORMINAL | NORMINAL | LOW |
| Product | RCPX | NORMINAL | NORMINAL | NORMINAL |
| | RUSE | LOW | HIGH | HIGH |
| Project | FCIL | HIGH | HIGH | HIGH |
| | SCED | NORMINAL | NORMINAL | NORMINAL |

개발 분야의 생산성 측정 단위에는 Early Design Model의 개인, 플랫폼, 제품, 프로젝트에 대한 각 Driver들의 연차별 목표를 설정하였으며, 이에 대한 생산성 계산은 SoftstarSystem의 Costar 7.02 프로그램을 적용하여 도출하였다. 유지보수 분야의 생산성 측정 단위는 MBASE/RUP Model의 개인, 플랫폼, 제품, 프로젝트에 대한 각 Driver들의 연차 별 목표를 설정하였으며, 이에 대한 생산성 계산또한 SoftstarSystem의 Costar7.02 프로그램을 적용 하였다.

KTDS의 유지보수 Coststar7.02 프로젝트 Cost Drivers for Component Sampling의 Drivers 값을 [표 3]과 같이 적용하였을 때 도출된 값은 Effort(PM) 20.1 , Duration (Mo) 10.1, Cost(K$) 9.4 , Productivity 818.8 로 적용되었으며, 입력한 Size 16472에 대하여 Unadjusted Function Point 259, Adjusted Function Point 311이 도출되었다.

[표. 3] KTDS MBASE/RUP에 의한 Cost Driver

| DRIVER GROUP | Driver | '09(e)년 | '10년 | '11년 |
|---|---|---|---|---|
| Personal | ACAP | HIGH | VERY HIGH | VERY HIGH |
| | APEX | HIGH | VERY HIGH | VERY HIGH |
| | PCAP | VERY HIGH | VERY HIGH | VERY HIGH |
| | PLEX | HIGH | VERY HIGH | VERY HIGH |
| | LTEX | HIGH | VERY HIGH | VERY HIGH |
| | PCON | HIGH | VERY HIGH | VERY HIGH |
| Platform | TIME | HIGH | HIGH | NOMINAL |
| | STOR | NORMINAL | HIGH | HIGH |
| | PVOL | LOW | LOW | LOW |
| Product | RELY | NOMINAL | NOMINAL | NOMINAL |
| | DATA | HIGH | HIGH | HIGH |
| | CPLX | LOW | LOW | NOMINAL |
| | RUSE | LOW | HIGH | HIGH |
| | DOCU | VERY LOW | NORMINAL | VERY HIGH |
| Project | TOOL | NORMINAL | NORMINAL | VERY HIGH |
| | SITE | NORMINAL | NORMINAL | VERY HIGH |
| | SCED | LOW | NORMINAL | VERY HIGH |




[그림. 5] Function Point Setting Component

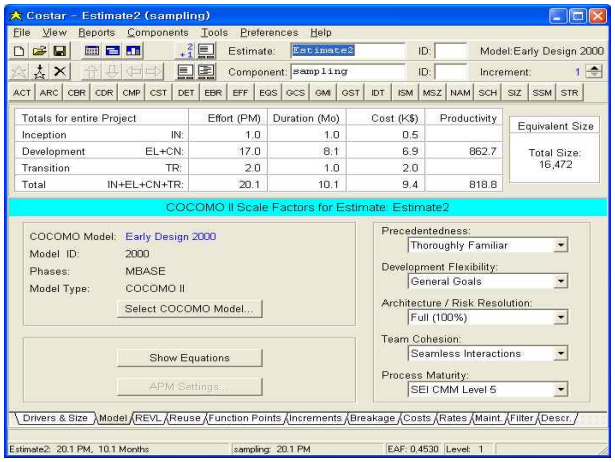
[표. 4] Function Point Setting For Component

| | Simple | Average | Complex |
|---|---|---|---|
| External Input | 4 | 4 | 2 |
| External Output | 3 | 4 | 2 |
| Logical Internal File | 3 | 3 | 3 |
| External Interface File | 3 | 2 | 3 |
| External Inquiry | 2 | 3 | 0 |

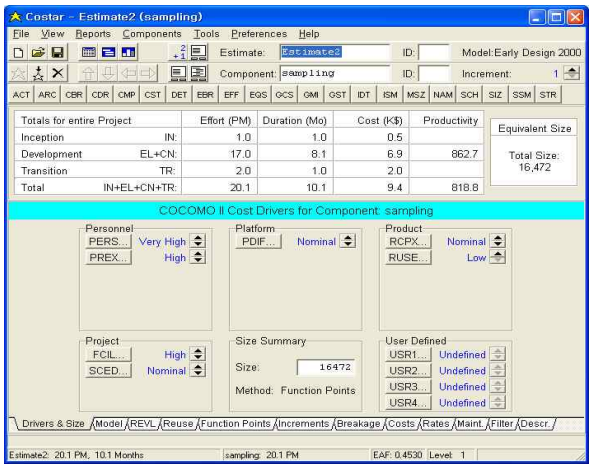
이에 따라 KTDS의 2009년 목표 생산성 측정을 위한 FP(Function Point)의 구성요소인 EI, EO, LIF, EIF, EI의 입력 값은 [표 4]와 같이 구성하였으며, 이에 따라 산출된 생산성관련 지표의 값은 [그림 5]과 같다.

산출된 값에 대한 Model은 Early Design 2000이 적용되었으며 Phases: MBASE에 Precedented 되는 Thoroughly Familiar로 설정하였으며, Development Flexibility는 General Goals, Architecture/Risk Resolution은 Full (100%), Team Cohesion : Seamless interactions로 설정하였다.

위 설정에 대한 지표 값은 Appendix를 참고하기 바라며, KTDS의 생산성 측정을 위한 표준 설정 값을 통한 Costar 입력 값은 [그림 6]과 같다.



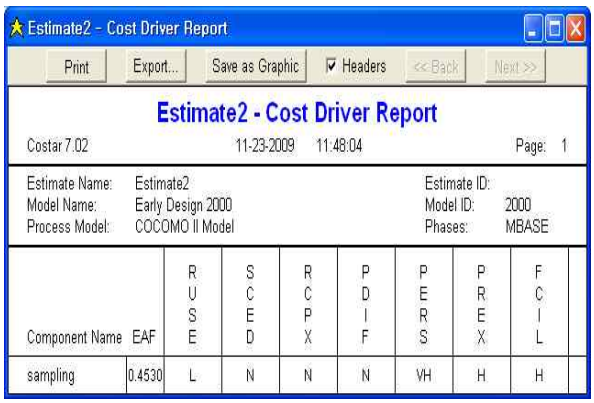

[그림. 6] COCOMO II Scale Factors for estimate



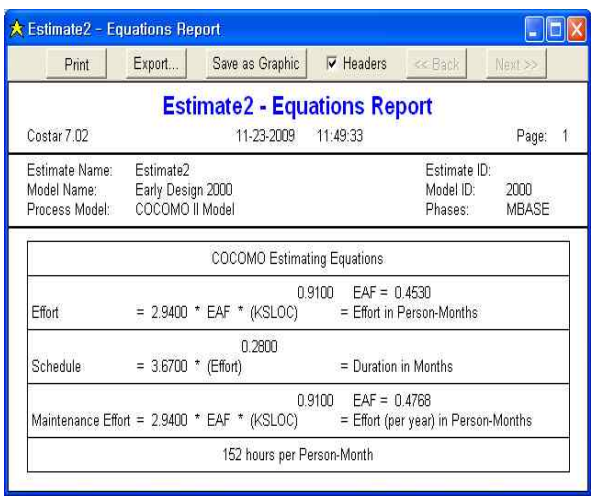

[그림. 7] Cost Driver For Component Sampling

Function Point Setting Component와 COCOMO II Scale Factors for estimate Setting을 적용하여 [표 2]와 같이 Cost Driver Setting을 [그림 7]과 같이 적용하여 COCOMO II Early design Model에 대한 생산성 지표를 도출할 수 있다.

입력된 COCOMO II 모델을 통해 KTDS 개발공수를 견적하였다. 7개의 입력인자와 5개의 선형인자를 적용하여 출력결과는 [그림 8], [그림 9]과 같다.




[그림. 8] Cost Driver Report




[그림. 9] Estimate Equations Report

[표. 5] KTDS 연도별 생산성 목표

| | '09년 | '10년 | '11년 |
|---|---|---|---|
| TOTAL | 20.1 | 15.3 | 13.3 |
| Productivity | 15.5 | 20.3 | 23.4 |

상기 도출된 Costar Effort(PM)과 AFP(Adjusted Function Point)를 사용하여 KTDS의 전사 생산성 지표를 도출한다.

$$\text{Productivity} = \frac{\text{Adjusted Function Points}}{\text{Effort(PM)}}$$

연도별 생산성 목표는 COCOMOⅡ 2000 Early Design Model에 의한 생산성 관리체계 및 연차별 목표설정으로 적용되었으며, 어플리케이션 유형, 직원역량, 플랫폼 난이도, 재사용성 등의 [표 2]의 Driver들의 내용에 따라 적용하였다.

## 5. 결 론

본 논문은 한국 IT산업 환경을 고려한 다양한 COCOMO Ⅱ 모델의 개발, 유지보수에 따라 설계된 생산성 측정 및 관리에 대한 지표를 기준으로 적용해보았으며, KTDS의 데이터를 기반으로 보정한 생산성 목표는 2009년 15.5 , 2010년 20.3, 2011년 23.4의 정량적 관리를 가능하게 한다. 또한, 본 논문은 견적과 실행계획의 연결고리로서 COCOMO Ⅱ 생산성 표준을 정립하였고, 객관적인 생산성 목표 수립이 가능함을 확인할 수 있었다.

본 논문의 연구를 제시된 보다 정밀한 데이터 조사 및 분석을 통해 보정된 COCOMO Ⅱ 모델의 정확도를 높이고, 실질적인 표준 WBS 체계와 표준공수가 구축되어 실행계획수립이 용이해지길 기대해본다.

표준 WBS의 정립을 위해서는 정량적인 FP산정과 플랫폼 이슈, 정확한 MM 산출이 선행되어야 한다. 하지만 현재 IT산업의 투입인력계획대비 실제 투입공수의 Gap이 현실적으로는 큰 차이가 있으며, 이를 내부적으로 관리하기 위한 COCOMOⅡ 표준 생산성 지표와 연계된 관리계획이 진행되어야 하고, 산출물 관점에서 견적한 각종 파생 Activity와 Task를 공정의 관점에서 재정리하여 나열해야 한다. Task와 연계된 모든 Role에 공수가 이미 분배되어 있기 때문에, 여기에 자원투입율을 부여하면 Task, Activity, 단계로의 상향식으로 공수 및 소요일을 도출할 수 있다. 다음으로 프로젝트 시작일을 설정하고 모든 Activity와 Task의 선후관계를 결정하고 나면 최종 일정계획을 도출할 수 있다. 마지막으로 COCOMOⅡ 모델의 Cost Drivers의 한국형 Customizing 작업과 더불어 Appendix의 기준에서 보다 세밀하고, 실질적인 모델설계를 기대한다.

## 6. 참 고 문 헌

[1] Barry W. Boehm, "Software Cost Estimation with COCOMOⅡ", Prentice Hall, 2000

[2] Maxwell, K". Applied Statistics for Software Managers", Prentice Hall, New Jersey, 2002.

[3] Barry Boehm, Ricardo Valerdi, Eric Honour, "The ROI of Systems Engineering: Some Quantitative Results for Software Intensive Systems", System Engineering, Wiley Periodicals 2008

[4] David Yu, "A Customizing Case of COCOMO Model", KPMA, Symposium & Seminar 2006

[5] Vu Nguyen, Barry Boehm, "A Constrained Regression Technique for COCOMO Calibration", Empirical Software Engineering and Measurement Conference, 2008

[6] (사)대한사업공학회, "소프트웨어 유지보수 대가기준 모형 연구", 한국전산원, 2004.10

## 저 자 소 개

**박 철 구**

순천대학교 산업대학원 정보통신공학과에서 석사 학위 취득, 현재 명지대학교 산업공학과 박사 과정. 현재 KT 데이터 시스템스에서 부장으로 재직 중.
관심분야는 Data Quality, Data Productivity 등이다.

주소: 서울 양천구 목동 924번지

**김 창 은**

TEXAS A&M 석, 박사학위 취득, 현재 명지대학교 산업경영공학과 교수로 재직 중.
관심분야는 CMMS, TPM, ERP, Six-Sigma, CALS/.EC 등이다.

주소: 경기도 용인시 처인 구 남동 산 38-2 명지대학교 산업경영공학과 제1공학관 537호