개략적 일정추정표 기반 소프트웨어 개발노력과 일정 추정

Estimation of Software Development Efforts and Schedule Based on A Ballpark Schedule Estimation Table

박 영 목*
Young-Mok Park

요 약

프로젝트 관리자는 입찰 또는 개발을 성공하기 위해 비용과 일정을 프로젝트 초기 단계에서 보다 정확히 추정해야만 한다. 평균적으로 대부분의 프로젝트들이 수행하고 있는 명목상의 개발기간은 경험법칙, 선형함수 추정경험 또는 개략적 일정 추정표로부터 유도할 수 있다. 명목상 개발기간에는 다양한 경험법칙이 존재하며, 선형함수 모델은 충분한 정보를 제공하지 못하여 특정 규모의 프로젝트에 적합한 개발노력과 일정을 결정하기 어렵다. 본 논문은 개략적 일정 추정표로부터 개발노력과 일정을 유도하는 통계적 회귀분석 모델을 제시하였다. 먼저, 개략적 일정 추정표에서 제시하고 있는 최대 단축기간, 효율적 개발기간과 명목상 개발기간을 재정의 하였다. 다음으로 개발노력과 기간과의 관계를 고찰하고, 개략적 일정 추정표에 제시되지 않은 특정 규모의 소프트웨어에 대한 개발노력 추정 모델과 개발노력에 따른 일정을 보다 정확히 추정할 수 있는 모델을 제시하였다. 제시된 회귀분석 모델은 기존의 방법들과 비교하여 상대오차를 최대 2%까지 줄이는 효과를 얻었다. 또한, 특정 규모의 소프트웨어에 대해서도 개발노력과 일정을 추정할 수 있었다.

Abstract

In order to succeed in a bid or development, the project manager should estimate its cost and schedule more accurately in the early stage of the project. Usually, the nominal schedule of most projects can be derived from rule of thumb, first-order estimation practice, or ball-park schedule estimation table. But the rule-of-thumb models for the nominal schedule estimation are so various, and the first-order estimation practice does not provide sufficient information. So they do not help much to decide on the proper development effort and schedule for a particular size of project. This paper presents a statistical regression model for deciding the development effort and schedule of a project using the ball-park schedule estimation table. First, we have redefined such words suggested in the ballpark schedule estimation table as shortest possible schedule, efficient schedule and nominal schedule. Next, we have investigated the relationship between the development effort and the schedule. Finally, we have suggested a model for estimating the development effort and the more accurate schedule of such particular sizes of software as are not presented in the ball-park schedule estimation table. The experimental results show that our proposed regression analysis model decreases the mean magnitude of relative error by 2% at maximum. Also, this model can estimate the development effort and schedule for a particular size of software.

☞ Keyword : 일정(Schedule), 개략적 일정 추정표 (Ballpark Schedule Estimation Table), 최단 개발기간(Shortest Possible Schedule), 효율적 개발기간(Efficient Schedule), 명목상 개발기간 (Nominal Schedule)

1. 서 론

프로젝트 개발 초기 단계에서 소프트웨어를 어느 정도의 일정 (Schedule)으로 개발할 수 있는지 정확히 추정하기는 현실적으로 불가능하다. 어떤 조직은 요구사항 분석 단계를 수행하기 전에 총 비용 (Cost)의 ±10% 이내에서 비용 추정 결과를 명문화하기를 원한다.[1] 고객이 원하는 것이 무엇인지 확신할 수 없는 개발 초기단계에서 이 범위 수준의 추정 정밀도는 이론적으로 불가능하며, 대부분의 소프트웨어 프로젝트들은 추정된 일정이 25%에서 100%를 초과하고 있다.[1,2] 그러나

* 정 회 원 : 진주산업대학교 학술정보전산팀장
ympark@jinju.ac.kr
[2007/03/14 투고 - 2007/03/30 심사 - 2007/06/27 심사완료]

사업은 사업이다. 프로젝트 관리자는 입찰에 참여하기 위해 또는 주어진 비용과 일정 내에서 성공적인 개발을 위해 사업 수행 여부 결정에 필요한 비용과 일정을 프로젝트 초기 단계에서 추정해야만 한다.[3]

소프트웨어의 규모 (Size), 비용과 일정의 추정 정확도는 개발될 소프트웨어 자체에 의존한다. 즉, 소프트웨어는 복잡성, 변경 가능성, 비가시성의 특징으로 인해, 개발이 진행되면서 점점 더 상세화되는 과정을 거친다. 이에 따라 추정도 프로젝트 초기 시점에서는 개략적인 점 추정 (Point Estimation) 결과를 얻고, 추정 오차를 반영하여 구간 추정 (Interval Estimation) 결과를 얻는다. 이후, 개발이 진행되면서 추정 결과 또한 점진적으로 정교 (Refinement)해지는 과정을 거친다.[1,4,5] 결국, 프로젝트 초기의 계획단계에서 대부분의 프로젝트 관리자들이 찾고자 하는 것은 특정 규모의 프로그램을 얼마의 기간 내에 개발할 수 있는지에 대한 정확한 값이 아니라 개략적인 추정 값이다.

개발 일정에 대해 가장 실현성이 있는 점 추정 결과를 어떻게 얻을 수 있는가? 이는 프로젝트 계획에서 가장 중요하면서도 어려운 부분이 될 수 있다.[2] 왜냐하면 대부분은 경쟁에서 이기기 위해, 입찰에 성공하기 위해 또는 경험 부족으로 인해 비현실적인 단기간의 일정을 제시하기 때문에 IT 분야의 프로젝트 성공률이 30%를 넘지 못하고 있다.[6]

프로젝트의 개략적인 일정을 결정하기 위한 많은 연구가 되고 있다. 먼저, 프로젝트 일정을 구분하는 연구로 Peters[7], Mulherji[8], Roetzheim[9], Cukie[10]과 Khaiter[11]이 있다. 프로젝트 일정은 개발이 불가능한 영역 (Impossible Region), 최소의 개발기간을 목표로 하는 최대로 단축할 수 있는 개발기간 (Shortest Possible Schedule, SPS), 최소의 개발비용을 목표로 하는 효율적인 (또는 최적의) 개발기간 (Efficient Schedule, ES)과 평균적으로 대부분의 프로젝트들이 수행하고 있는 명목상의 개발기간 (Nominal Schedule, NS)으로 구분한다. 그러나 ES와 NS의 개념 정의에도 다양한 이견이 존재하여 혼란을 겪고 있다.

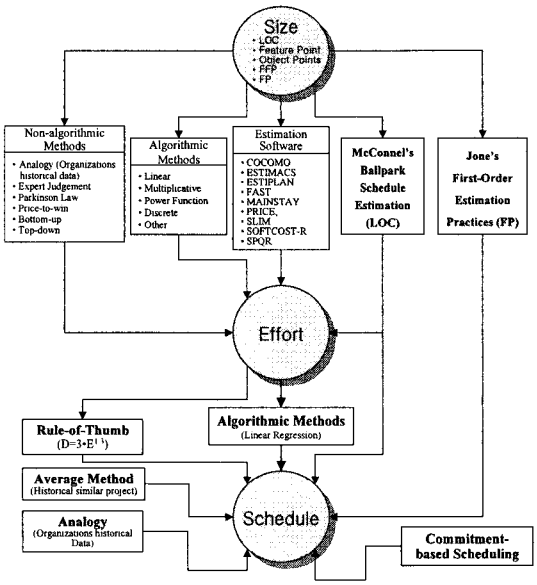
규모에 기반하여 노력 (Effort)이 추정되며, 노력에 기반하여 NS가 결정된다. NS를 결정하는 방법으로는 일반적으로 경험 법칙[12-34], Jones[2]의 선형함수 추정 경험 (First-order Estimation Practices)과 McConnel[24]의 개략적 일정 추정표 (Ballpark Schedule Estimation Table)를 사용할 수 있다. 그러나 경험 법칙은 다양한 모델이 존재하고 있다. 따라서 특정 프로젝트에 적합한 모델이 어떤 것인지에 대한 기준이 없는 실정이며, 선형함수 추정 경험은 기능점수 (Function Point, FP)에 기반하고 있으나 너무나 단순하여 원하는 정보를 충분히 얻지 못한다.

본 논문은 McConnel[24]의 개략적 일정 추정표에서 제시되고 있는 SPS, NS와 ES에 대한 정의를 명확히 재정립하여 적용의 오류를 방지한다. 또한, 개발일정을 보다 정확히 추정하기 위한 모델을 제시한다. 2장에서는 일정 추정에 있어 다양한 방법들을 살펴보고 각 방법이 내포하고 있는 문제점을 고찰하여 본다. 3장에서는 먼저, 개략적 일정 추정표에 기반하여 개발 일정에 대한 용어를 재 정의하여 용어 혼란을 방지하고자 한다. 다음으로, 개발 노력과 기간의 관계를 고찰하여 본다. 마지막으로 특정 규모의 소프트웨어에 대한 개발노력과 일정을 보다 정확히 추정할 수 있는 모델을 제시한다.

2. 일정 추정 관련연구와 문제점

2.1 추정 방법과 선호도

소프트웨어 추정 순서와 방법은 Leung와 Fan[35], Johnson[36], Mildred[37], Molϕkken과 Jϕrgensen[38]와 Gruenwalt[39]의 연구 결과를 종합하면 그림 1과 같다.



(그림 1) 소프트웨어 추정 순서

일반적으로 개발 노력과 일정은 알고리즘 방법으로 각각 식 (1)과 식 (2)로 구해진다. 여기서 Productivity Factor는 1명이 1개월 동안 개발할 수 있는 프로그램의 규모이다.

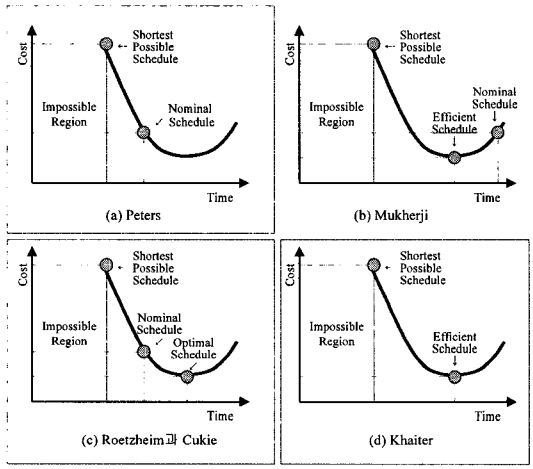
$$Effort(E) = Productivity\ Factor \cdot Size^{Penalty} = a \cdot S^{b} \quad (1)$$

$$Time(D) = a \cdot Effort^{Penalty} = a \cdot E^{b} \quad (2)$$

소프트웨어 개발과정에서 프로젝트 관리자들이 어떤 추정 방법들을 선호하고 있는지에 대해 Mol økken과 Jørgensen[38]가 조사한 결과를 살펴보면, 아직까지도 분석이나 실험에 기반을 둔 통계적 모델 (소프트웨어 비용추정 모델)을 13~26%만 적용하며, 대부분은 전문가 자문이나 직관과 경험 또는 유사한 프로젝트들로부터 유추하여 추정하는 방법을 선호하고 있다.

2.2 개발일정 형태

개발일정에 대해서는 Peters[7], Mulherji[8], Roetzheim[9], Cukie[10]과 Khaiter[11]의 연구 결과가 있으며, 이를 그림 2에 제시하였다. 그림에서 알 수 있듯이 개발일정은 다양한 용어들로 설명된다. 본 절에서는 (b)를 제외하고 (a), (c)와 (d)만으로 해석하여 보자. (b)를 제외한 이유는 3.1 절에서 설명된다. (a), (c)와 (d)를 해석하면 소프트웨어 개발일정은 일반적으로 SPS, ES 또는 NS 중 하나로 선택될 수 있다. ES를 Optimal Schedule (OS)라 부르기도 한다.



(그림 2) 소프트웨어 개발기간과 비용 관계

모든 프로젝트는 일정을 무한정 단축시킬 수 없는 한계점인 SPS를 갖고 있다. SPS는 아무리 많은 인원을 추가하더라도 이 일정 이하로는 더 이상 단축시킬 수 없는 최소한의 개발기간이다. 이 한계점 이하를 불가능 영역이라 한다. 또한, 어떤 시점 이후는 추가적으로 인력을 추가하더라도 이전 보다 생산성이 저하되는 수익체감의 법칙 (The Law of Diminishing Return) 현상을 나타낸다. 이를 수익체감 점이라 하며, ES 또는 OS가 이에 해당된다.

고객은 보다 빠르게 (Faster), 보다 좋은 제품을 (Better) 보다 저렴한 가격으로 (Cheaper) 개발하기를 원한다. 그러나 현실적으로 이 3가지 제약조건을 모두 만족시키며 개발을 성공할 수는 없다. 개발일정은 프로젝트가 비용과 시간 중 어떤 것을

보다 중요시하는가에 따라 결정된다. 시간을 보다 중요시하면 최소 시간이 소요되는 SPS를 선택할 수 있으며, 비용을 보다 중요시하면 최소 비용이 소요되는 ES를 선택할 수 있다.

현실적으로 많은 개발 상황에서 비용 보다는 시간이 보다 중요시되는 경우가 대부분이다. 따라서 대부분의 프로젝트들은 현실적으로 ES로 개발을 수행하지 못하며, 개발팀의 능력과 개발환경과 도구가 이상적으로 지원되지 못하면 SPS로도 개발을 하지 못한다. 결국, 대부분의 프로젝트들은 SPS와 ES의 사이에 위치한 NS를 택해 개발하게 된다. NS는 보다 일정을 단축시키거나 연장시킬 수 있으며, 고객의 보다 빠른 개발 요구로 인해 일정 연장은 거의 고려하지 않고 단축에만 많은 관심을 갖고 있다.

2.3 개발일정 추정 방법

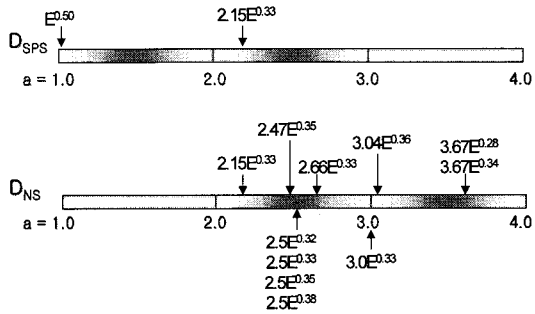
소프트웨어 추정과정은 한번에 결정되는 것이 아니라 계속적으로 진행되는 정제 과정을 거친다. 따라서 일정도 처음에는 개략적인 점 추정을 하고, 추정 승수를 적용하여 구간추정 정보를 제공하고 프로젝트가 진행되면서 정확도를 증가시키기 위해 주기적으로 구간을 정제한다.

2.3.1 점 추정

소프트웨어 개발일정을 확정짓기 위해 가장 중요한 일정이 NS이다. 왜냐하면 이 일정을 알아야만 일정 단축과 연장 여부를 결정할 수 있기 때문이다. 그러면 NS를 어떻게 점 추정으로 결정할 수 있는가? 의미 있는 소프트웨어 일정을 결정하기위해 가장 현실적으로 부닥치는 장애물중 하나는 소프트웨어 일정에 대해 유용하고 구체적인 정보를 얻기가 어렵다는 점이다. 가장 쉽게 얻을 수 있는 정보는 개발조직의 과거 이력 데이터로부터, 추정 소프트웨어 사용 또는 경험법칙을 거론하고 있는 논문이나 책에서 얻는 것이다. 그러나 새로 개발될 소프트웨어와 유사한 과거 개발 이력 데이터들이 많지 않으며, 추정 소프트웨어는 구입에 고가의 비용이 소요되며, 관련 논문과 책을 모두 찾는 것은 많은 시간이 소요되어 이들 방법은 좋은 결과를 얻지 못한다. 차선책으로 얻을 수 있는 정보로는 규모가 FP로 추정된 경우 선형함수 추정 경험[2] 사용, LOC (Lines-of-code)인 경우 개략적 일정 추정표[24] 사용 방법이 있다. 본 절에서는 경험법칙, Jones[2]의 선형함수 추정 경험과 McConnel[24]의 개략적 일정 추정표 (이후부터 간략히 "일정 추정표"라 한다.) 방법을 살펴본다.

2.3.1.1 경험법칙 적용

개발일정은 일반적으로 노력 (E)에 기반하여 추정되며, 대부분은 경험법칙으로 $D = 3.0 \cdot E^{1/3}$을 적용한다. 당신의 개발조직에서 어떤 보다 정확한 이력데이터를 갖고 있지 않다면 경험법칙 공식을 사용하면 좋은 시발점이 될 수 있다. 기존에 제시된 SPS, NS, ES와 관련된 연구 결과를 요약하여 그림 3에 제시하였다.[12-33,43]



(그림 3) a 값 기준 개발일정 추정 모델 분포

SPS는 일반적으로 $D_{SPS} = \sqrt{E}\ (= E^{0.50})$ (Square Root Law)을 적용한다[12]. NS는 일반적으로 $D_{NS} = a \cdot \sqrt[3]{E}\ (= aE^{0.33})$ (Cube Root Law)를 적용하고 있다[12]. $3.0E^{0.33}$에 대해, 대부분은 NS를 설명[12,18,28-33]하고 있으나 Richard[43]만 ES를

거론하고 있다. 따라서 $3.0E^{0.33}$과 $2.15E^{0.33}$를 NS로 해석함이 타당할 것으로 판단된다. 즉, SPS는 $E^{0.50}$으로, NS는 $a \cdot E^{b}$ $(2.0 \leq a \leq 4.0)$로 해석하면, 아직까지 ES에 대한 경험법칙은 제시되지 않은 것으로 해석이 될 수 있다. 이와 같이 경험법칙은 SPS, NS와 ES에 대해 용어의 혼돈을 일으키고 있으며, 개발될 특정 규모의 소프트웨어에 대한 NS를 결정시 어떤 모델을 적용하는 것이 가장 적합한지 판단할 수 없는 문제점을 갖고 있다.

2.3.1.2 선형함수 추정 경험 적용

기능점수에 기반하여 Jones[2]은 $D = FP^{a}$ 수식에 대한 표 12의 값을 적용하여 개발기간을 추정하는 모델을 제시하였다. 만약 FP로 규모를 추정하였다면 Jones[2]의 선형함수 추정 경험 표를 적용하여 개략적인 일정을 얻을 수 있다. 이 방법은 보다 면밀한 일정 추정의 대안이 되지 못하나 추측하는 것 보다는 좋은 개략적인 일정 추정을 얻는 단순한 방법을 제공한다.

(표 1) 기능점수 기반 개발기간 추정 모델

구분	a		
	Best	Average	Worst
System Products	0.43	0.45	0.48
Business Products	0.41	0.43	0.46
Shrink-Wrap Products	0.39	0.42	0.45

실제로 우리가 얻고자 하는 것은 특정 규모(Size)의 소프트웨어 개발에 소요되는 노력과 시간이 얼마이고, 개발 기간을 얼마까지 단축시킬 수 있으며, 이때 소요되는 노력은 얼마인지이다. 이 정보에 기반을 두고 비용-일정 타협을 수행할 수 있다. 그러나 선형함수 추정 경험은 소프트웨어 규모와 프로젝트 분류 정보를 갖고 있지 않아 효율적인 적용이 되지 못한다.

2.3.1.3 일정 추정표 적용

일정 추정표는 개발 조직의 과거 이력 데이터가 충분하지 못할 때 사용된다. McConnel[24]은 표 2와 같이 LOC에 기반하여 개략적인 일정을 추정하는 표를 제시하였다.[3,4,24] 여기서, Size를 S로, Effort를 E로, Schedule (Time)을 D로 정의하자. 여기서 단위는 S는 KLOC, E는 Person-Months (PM), D는 Months (M)이다. 이 표에 있는 데이터의 정확도는 특정 개발조직의 과거 이력 데이터들로부터 얻은 정확도와 조금도 일치하지 않을 수 있다. 만약 당신의 조직이 과거 프로젝트의 규모를 기록하지 않았거나 유사한 프로젝트들이 없다면 이 표를 이용하면 직감에 의한 방법 보다 정확한 값을 얻을 수 있다.

McConnel[24]의 일정 추정표에는 10 KLOC이하의 데이터는 제시되어 있지 않다. 왜냐하면 이 이하의 소프트웨어 규모는 팀 단위가 아닌 1명이 충분히 개발할 수 있는 것으로 생각하였기 때문이다. 이후, Thomas[4]가 NS에 대한 개발노력과 일정을 추가로 제시하였기 때문에 참고로 표에 추가하였다.

이 표는 개념정의 단계부터 요구사항 분석단계까지 소요되는 일정과 노력은 생략되어 있으며, 요구사항 분석 단계가 완료된 상태에서 요구사항 명세서에 기반하여 아키텍쳐 설계 단계부터 시스템 시험단계 까지 소요되는 노력과 일정이다. 따라서 실제 적용시 프로젝트 수행에 투입되는 총 노력과 일정을 얻기 위해서는 개념정의와 요구사항 분석단계가 차지하는 비율을 고려하여야만 한다.

특정한 규모의 프로젝트를 개발하고자 할 때, 표 3에서 개발팀의 능력과 개발환경을 평가하여 ES와 NS중 어느 것이 가장 적합한지를 선택한 후 표 2를 적용하여야 한다.

(표 2) LOC 기반 일정 추정표

구분	System Products						Business Products						Shrink-Wrap Products					
	SPS		ES		NS		SPS		ES		NS		SPS		ES		NS	
S	D	E	D	E	D	E	D	E	D	E	D	E	D	E	D	E	D	E
3	-	-	-	-	-	-	-	-	-	-	2	3	-	-	-	-	3	5
5	-	-	-	-	-	-	-	-	-	-	3	5	-	-	-	-	4	6
6	-	-	-	-	-	-	-	-	-	-	4	6	-	-	-	-	6	9
10	6	25	8	24	10	48	3.5	5	4.9	5	6	9	4.2	8	5.9	8	7	15
15	7	40	10	38	12	76	4.1	8	5.8	8	7	15	4.9	13	7	12	8	24
20	8	57	11	54	14	110	4.6	11	7	11	8	21	5.6	19	8	18	9	34
25	9	74	12	70	15	140	5.1	15	7	14	9	27	6	24	9	23	10	44
30	9	110	13	97	16	185	5.5	22	8	20	9	37	7	37	9	32	11	59
35	10	130	14	120	17	220	5.8	26	8	24	10	44	7	44	10	39	12	71
40	11	170	15	140	18	270	6	34	9	30	10	54	7	57	10	49	13	88
45	11	195	16	170	19	310	6	39	9	34	11	61	8	66	11	57	13	100
50	11	230	16	190	20	360	7	46	10	40	11	71	8	79	11	67	14	115
60	12	285	18	240	21	440	7	57	10	49	12	88	9	96	12	83	15	145
70	13	350	19	290	23	540	8	71	11	61	13	105	9	120	13	100	16	175
80	14	410	20	345	24	630	8	83	12	71	14	125	10	140	14	120	17	210
90	14	480	21	400	25	730	9	96	12	82	15	140	10	170	15	140	17	240
100	15	540	22	450	26	820	9	110	13	93	15	160	11	190	15	160	18	270
120	16	680	23	560	28	1,000	10	140	14	115	16	200	11	240	16	195	20	335
140	17	820	25	670	30	1,200	10	160	15	140	17	240	12	280	17	235	21	400
160	18	960	26	709	32	1,400	11	190	15	160	18	280	13	335	18	280	22	470
180	19	1,100	28	910	34	1,600	11	220	16	190	19	330	13	390	19	320	23	540
200	20	1,250	29	1,300	35	1,900	12	250	17	210	20	370	14	440	20	360	24	610
250	22	1,650	32	1,300	38	2,400	13	330	19	280	22	480	15	580	22	470	26	800
300	24	2,100	34	1,650	41	3,000	14	420	20	345	24	600	16	725	24	590	29	1,000
400	27	2,900	38	2,350	47	4,200	15	590	22	490	27	840	19	1,000	27	830	32	1,400
500	30	3,900	42	3,100	51	5,500	17	780	25	640	29	1,100	20	1,400	29	1,100	35	1,800

(표 3) 개발일정 선정 기준

구분	SPS	ES	NS
고객 요구	Time-to-market (최소 일정)	Project Cost (최소 비용)	Nominal (-)
프로젝트 목표	Rapid Development	Efficient Development	Nominal Development
개발팀 구성	Best (Top 10%)	Better (Top 25%)	Nominal (Top 50%)
인원 추가 효과	일정 지연	일정 단축 가능	일정 단축 가능

2.3.2 구간 추정

소프트웨어의 각 개발 단계별 규모, 노력과 일정 추정 오차는 표 4에 제시되어 있다.[14]

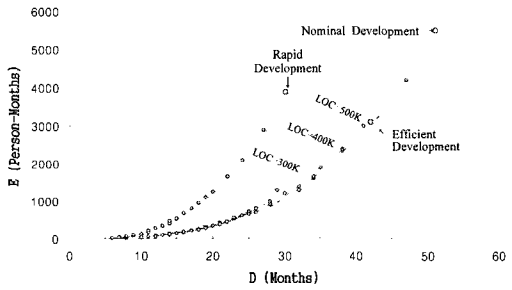
(표 4) 개발단계별 규모, 노력과 일정 추정 오차

단계	규모와 노력 추정 오차 (%)		일정 추정 오차 (%)	
	낙관적	비관적	낙관적	비관적
초기 개념 단계 (Draft Vision Statement)	25	400	60	160
개념 승인 단계 (Baselined Vision Statement)	50	200	80	125
상세 요구사항 정의 완료 (Requirement Spec)	67	150	85	115
설계 규격서 (Architecture Design) 완료	80	125	90	110
상세설계 완료 시점	90	110	95	105
개발 종료 시점	100	100	100	100

3. 일정 추정표 기반 노력과 일정 추정 모델 제안

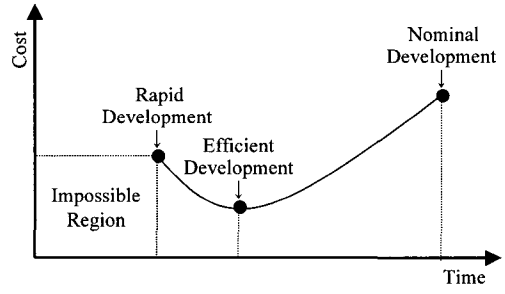
3.1 개발일정 개념 재정립

표 2 McConnel[24]의 일정추정표에 대해 System Products의 규모별로 개발일정에 따른 노력을 그림 4에 제시하였다. 여기서, Shortest Possible Schedule을 Rapid Development (RD)로, Efficient Schedule을 Efficient Development (ED)로, Nominal Schedule을 Nominal Development (ND)라 부르자.



(그림 4) System Products의 개발기간에 따른 노력

Business Products와 Shrink-Wrap Products도 동일한 형태를 갖고 있다. 그림 4의 결과로부터 개발 노력은 Mukherji[8]이 제시한 그림 2 (b) 형태가 아니라 그림 5와 같이 $E_{ED} < E_{RD} < E_{ND}$ 관계가 있음을 알 수 있다. 그림 5의 결과를 Mukherji[8] 개발일정 관계로 혼돈하지 말기 바란다.



(그림 5) 개발기간에 따른 실질적인 비용 관계

McConnel[24]의 일정 추정표를 이용하여 개발노력과 개발기간의 비율을 분석한 결과는 표 5에 제시되어 있다. RD는 ED에 비해 일정은 0.70으로 적게, 노력은 1.14 ~ 1.17배 더 소요된다. ND는 ED에 비해 일정은 1.18 ~ 1.21배 더 소요되며, 노력은 1.76 ~ 1.85배 더 소요된다. 또한, RD에 비해 일정은 1.70 ~ 1.74배 더 소요되며, 노력은 1.51 ~ 1.59배 더 소요된다.

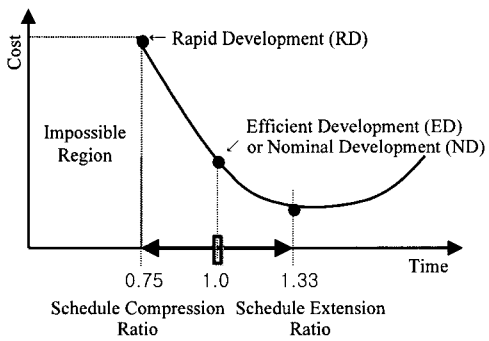
(표 5) 개발기간과 노력 비율

구분		*RD/ED*		*ND/ED*		*ND/RD*	
		시간	노력	시간	노력	시간	노력
System Products	최소	0.67	0.96	1.17	1.46	1.64	1.41
	평균	**0.70**	**1.17**	**1.21**	**1.85**	**1.74**	**1.59**
	최대	0.75	1.35	1.27	2.04	1.82	1.93
Business Products	최소	0.66	1.00	1.10	1.71	1.51	1.41
	평균	**0.70**	**1.14**	**1.18**	**1.78**	**1.70**	**1.57**
	최대	0.75	1.22	1.29	1.93	1.83	1.91
Shrink-Wrap Products	최소	0.67	1.00	1.11	1.64	1.57	1.29
	평균	**0.70**	**1.17**	**1.20**	**1.76**	**1.71**	**1.51**
	최대	0.78	1.27	1.30	2.00	1.86	1.88

Mukherji[8]가 제안한 그림 2의 (b) 일정은 (a), (c)와 (d)의 일반적인 개념의 일정 구분과 동일한 용어를 사용하고 있지만 개발기간에는 차이가 발생한다. 또한, 표 2의 McConnel[24]의 일정 추정표에서 SPS, ES와 NS의 일정 (D)만을 살펴보면 Mukherji[8] 일정 구분과 동일한 개발기간이 된다. 그러나 개발노력에는 차이가 발생한다. 여기서, 우리는 McConnel[24]의 일정 추정표에서 거론하고 있는 SPS, ES와 NS를 Mukherji[8]의 일정 구분으로 잘못 해석할 수 있다. 그러나, McConnel[24]의 일정 추정표에서 SPS, ES와 NS는 일반적으로 알고 있는 일정 구분과 전혀 다른 개념을 적용하고 있다. 왜냐하면 표 5의 결과를 종합하여 보면 Drinka[3]와 McConnel[24]이 제안한 SPS, ES와 NS

는 개발팀의 업무수행 능력, 개발환경과 도구에 기반하여 달성할 수 있는 일정임을 알 수 있다. 따라서 Mukherji[8]는 McConnel[24]의 일정 구분을 잘못 해석한 것으로 판단된다.

McConnel[24]의 RD, ED와 ND를 일반적으로 알고 있는 용어인 SPS, NS, ES로 어떻게 대응시킬 수 있는가? 먼저, RD는 SPS로 대응시킬 수 있다. 다음으로, ED와 ND는 프로젝트가 추구하는 일정 달성 목표로 그림 6과 같이 우리가 일반적으로 찾고자 하는 NS로 대응시켜보자.



(그림 6) 프로젝트 목표에 따른 개발일정 정의

이와 같이 해석하는 이유는 첫째로, SPS는 더 이상 단축시킬 수 없지만 NS와 ES는 25%까지 일정을 단축시킬 수 있기 때문이다. 둘째로, PRICE-S, SEER-SEM과 COCOMO-II 추정 소프트웨어는 NS의 일정 단축과 연장에 대해 동일한 개념을 적용하고 있다. 단지, SLIM과는 차이가 발생한다.[44] SLIM은 SPS를 기준(1.0)으로 130% 범위 내에서 실제적인 타협 영역 (Practical Tradeoff Region)을 설정하고 있기 때문이다.

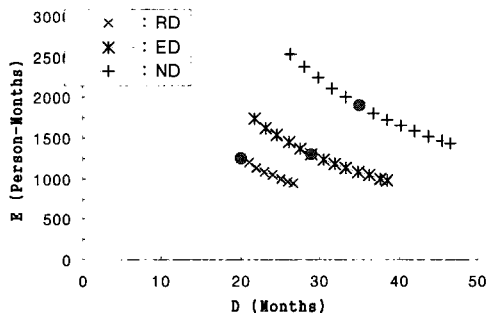
ED와 ND를 NS로 해석하는 것이 타당한지 검증하기 위해 일정 단축과 연장을 분석하여 보자. 일정 단축시 소요되는 노력은 식 (3)으로 구해진다. 여기서, SCF는 일정 단축 인자 (Schedule Compression Factor)이다.

SCF = 원하는 일정/초기 일정 단축된 일정에 소요되는 노력 = 초기 노력/SCF (3)

일반적으로 D_{NS}을 단축시킬 수는 최대 범위는 25% ($75\%D_{NS}$)이며, 연장은 130% ($1.33D_{NS}$)를 적용하고 있다. 즉, $0.75D_{NS} \le D_{NS} \le 1.30D_{NS}$의 범위로 D_{NS}를 선정할 수 있다.[2,17,45] System Products의 200 KLOC에 대해 RD, ED와 ND의 5% 단위로 일정 단축과 연장 기준을 적용한 결과는 그림 7과 같다. 그림에서 "●"는 일정 추정표의 일정과 노력에 대응하는 값이다. 여기서, RD는 일정 단축이 더 이상 불가능하기 때문에 일정 연장 기준만 적용하였다.

ED의 능력을 갖고 있는 개발팀이 일정 단축을 하면 D_{RD}로 개발을 할 수 있으나 개발노력은 많이 추가됨을 알 수 있다. ND의 능력을 갖고 있는 개발팀이 일정 단축을 하면 ED의 일정으로 개발은 가능하나 노력은 급격히 증가됨을 알 수 있다. 이는 개발팀의 업무 수행 효율성이 낮은 상태에서 단기간에 개발을 수행함으로써 나타나는 현상이다.

ED와 ND에 대해 일정 단축과 연장에 따른 개발 노력 그래프가 일반적으로 알고 있는 SRS와 ES 사이의 곡선 형태를 표현하고 있어, ED와 ND는 그림 6과 같이 우리가 일반적으로 찾고자 하는 NS로 해석하는 것이 타당함을 알 수 있다.



(그림 7) 일정 단축과 연장에 다른 노력

3.2 개발 노력과 일정 추정 회귀분석 모델

기존에는 경험법칙을 적용하지 않을 경우, 대부분은 McConnel[24]의 일정 추정표를 기반으로 개발기간을 결정하고 있다. 즉, 소프트웨어 규모에 기반하여 이론적으로 제시된 통계적 모델이 없는 실정이다. 따라서 본 장에서는 통계적 기반의 회귀분석 모델을 제시한다. 만약 표 2 McConnel[24]의 일정 추정 표에 제시된 소프트웨어 규모가 아닌 경우 어떻게 노력과 일정을 추정할 것인가? 예로, 350 KLOC의 시스템 프로젝트를 개발한다고 가정하자. 표 2에서 이 소프트웨어가 적용될 수 있는 경우는 300 KLOC와 400 KLOC이다. 300 KLOC인 경우 명목상 개발일정(NS)는 41개월에 3,000 PM이며, 400 KLOC인 경우 47개월에 4,200 PM이다. 이 경우, 100,000 LOC의 간격에 개발기간은 6개월 차이, 노력은 1,200 PM 차이가 발생한다. 그러면 300 KLOC와 400 KLOC 중 어떤 것을 적용하는 것이 보다 타당한가? 이에 대한 명확한 해답을 얻기가 불가능할 것이다. 이와 같은 문제점을 해결하기 위해 본 절에서는 표 2 McConnel[24]의 일정 추정표를 회귀분석을 거쳐 소프트웨어 규모에 기반하여 개발노력과 개발기간을 추정한 모델을 제시한다. 제시된 모델은 표 6과 같다. 예로, System Products의 RD의 개발노력과 개발기간 추정 모델은 $E=1.2792S^{1.3034}$ 와 $D=2.3449S^{0.4049}$ 를 얻었으며, 이는 System Products의 RD란에 있는 모든 소프트웨어 규모 vs. 개발노력과 규모 vs. 개발기간 데이터를 회귀분석하여 얻었다.

$E=a\cdot S^b$에서 $0.25\le a\le 2.94$, $1.21\le b\le 1.32$를, $D=a\cdot S^b$에서 $1.37\le a\le 3.47$, $0.40\le b\le 0.42$ 관계를 나타내고 있다. 제시된 표의 모델을 적용하면 개발될 소프트웨어 규모에 따라 보다 정확한 노력과 일정을 얻을 수 있을 것이다.

다음으로 개발노력에 기반하여 개발기간을 추정하여 보자. 식 (2)의 $D=a\cdot E^b$에 따라 회귀분석을 이용하여 개발기간을 추정한 모델은 표 7에 제시되어 있다. System Products, Business Products와 Shrink-Wrap Products의 개발일정은 $2.11\le a\le 3.07$, $0.30\le b\le 0.36$을 얻을 수 있었다. 이 결과는 Simmons[27]의 $2.15\le a\le 3.04$, $0.32\le b\le 0.38$과는 약간의 차이를 보였다.

(표 7) 개발노력 기반 일정 추정 모델

구분		개발기간 추정 모델
System Products	RD	$D=2.1866E^{0.3095}$
	ED	$D=2.8930E^{0.3313}$
	ND	$D=2.7516E^{0.3376}$
Business Products	RD	$D=2.1064E^{0.3105}$
	ED	$D=2.9459E^{0.3263}$
	ND	$D=2.8416E^{0.3296}$
Shrink-Wrap Products	RD	$D=2.2613E^{0.2980}$
	ED	$D=3.0706E^{0.3174}$
	ND	$D=2.8004E^{0.3359}$

(표 6) 규모 기반 개발노력과 일정 추정 모델

구분		규모 기반	
		개발노력 추정 모델	개발기간 추정 모델
System Products	RD	$E=1.2792S^{1.3034}$ $(R^2=0.9977)$	$D=2.3449S^{0.4049}$ $(R^2=0.9970)$
	ED	$E=1.3411S^{1.2566}$ $(R^2=0.9973)$	$D=3.4769S^{0.4172}$ $(R^2=0.9986)$
	ND	$E=2.9444S^{1.2170}$ $(R^2=0.9995)$	$D=3.9581S^{0.4112}$ $(R^2=0.9993)$
Business Products	RD	$E=0.3933S^{1.3279}$ $(R^2=0.9972)$	$D=1.7048S^{0.3968}$ $(R^2=0.9954)$
	ED	$E=0.4187S^{1.2780}$ $(R^2=0.9983)$	$D=2.3186S^{0.4067}$ $(R^2=0.9976)$
	ND	$E=0.8846S^{1.2357}$ $(R^2=0.9992)$	$D=2.6880S^{0.4150}$ $(R^2=0.9977)$
Shrink-Wrap Products	RD	$E=0.2537S^{1.3056}$ $(R^2=0.9975)$	$D=1.3711S^{0.4062}$ $(R^2=0.9955)$
	ED	$E=0.2784S^{1.2544}$ $(R^2=0.9988)$	$D=1.9379S^{0.4096}$ $(R^2=0.9962)$
	ND	$E=0.5550S^{1.2278}$ $(R^2=0.9994)$	$D=2.3355S^{0.4033}$ $(R^2=0.9975)$

McConnel[24]은 소프트웨어 규모에 기반하여 개발 노력과 일정을 표로 제시하였지만 제안된 모델은 규모에 기반하여 노력과 일정도 추정할 수 있는 통계적 모델과 더불어 개발노력에 기반하여 일정도 추정할 수 있는 장점도 갖고 있다. 또한, 제안된 모델을 적용할 경우 McConnel[24]의 일정 추정표에 제시되지 않은 특정 규모의 소프트웨어에 대해서도 보다 정확한 개발노력과 일정을 추정할 수 있는 효과를 얻을 수 있다.

3.3 제안 모델 평가

제안된 모델의 적합성을 평가하기 위해 McConnel[24]와 Thomas[4]의 모델과 비교한다. Thomas[4]는 RD의 경우 1.9, ED의 경우 2.94를 거론하고 있으나 ND에 대해서는 제시된 값이 없다. 따라서 ND는 ED의 값을 적용하였다. McConnel[24]은 $D = K \cdot E^{1/3}$에서 K의 값은 개발 목적에 의존하며, $2.0 \le K \le 4.0$의 범위를 가진다고 제안하였다. McConnel[24]의 경험법칙을 적용하기 위해서는 RD는 2.0, ND는 3.0, ED는 4.0을 적용할 수 있는지는 의문이다. 따라서 일정 추정표에 대해 $K = \frac{D}{E^{0.33}}$에 따라 표 8과 같이 K 값을 구해 평균값을 적용한다.

(표 8) 일정 추정표의 K 값

K	System Products			Business Products			Shrink-Wrap Products		
	최소	평균	최대	최소	평균	최대	최소	평균	최대
RD	1.83	**1.94**	2.17	1.79	**1.94**	2.09	1.80	**1.94**	2.13
ED	2.72	**2.92**	3.01	2.77	**2.90**	3.17	2.75	**2.90**	3.20
ND	2.79	**2.89**	3.00	2.68	**2.84**	3.03	2.79	**2.89**	2.97

모델의 성능 평가 기준으로 소프트웨어공학 분야에서 일반적으로 적용하고 있는 MMRE (Mean Magnitude of Relative Error)를 활용하였다. 상대오차 (Relative Error, RE)는 $\frac{(\text{실측치} - \text{추정치})}{\text{실측치}} \times 100(\%)$로 구해진다. n개의 데이터가 있을 경우 제안된 모델의 성능을 평가하기 위해서는 MRE (Magnitude of the RE) $= |RE|$를 계산하고 다시 $MMRE$ (Mean MRE) $= \frac{1}{n}\sum_i MRE_i,\ i = 1, 2, \cdots, n$를 계산한다. $MMRE$가 적을수록 제안된 모델이 실제 데이터를 잘 표현한다고 볼 수 있다. 제안된 모델과 McConnel[24]의 모델, Thomas[4]의 모델과 일정 추정 모델의 성능을 비교한 결과는 표 9에 제시되어 있다. 표에서 제안된 회귀분석 모델이 대부분의 경우에 대해 우수한 결과를 얻음을 알 수 있다.

(표 9) 일정 추정 모델 성능 비교

구분		모델 성능 (MMRE)		
		제안 모델	McConnel[24] 모델	Thomas[4] 모델
System Products	RD	**3.12**	3.76	3.48
	ED	1.64	1.62	**1.60**
	ND	**1.21**	1.52	2.23
Business Products	RD	**2.60**	3.47	3.64
	ED	2.19	**2.17**	2.34
	ND	2.30	**2.29**	3.96
Shrink-Wrap Products	RD	**2.67**	4.84	4.69
	ED	**2.68**	2.86	3.38
	ND	**1.24**	1.41	1.97

4. 결론 및 향후 연구과제

개발 초기에 개발될 소프트웨어의 규모를 정확히 추정하는 것은 현실적으로 불가능하다. 그 결과, 추정된 규모에 기반하여 개발노력의 양을 추정하는 것 또한 부정확해진다. 마지막으로 개발노력에 기반하여 개발기간을 추정하는 것 또한 부정확한 결과를 얻는다. 이를 극복하기 위해서는 과거에 개발된 유사한 프로젝트 데이터를 충분히 확보하고 이에 기반하여 유추 해석하는 것이 최선의 방법이 될 수 있다. 그러나 충분한 데이터를

확보하는데도 한계가 있기 때문에 대부분의 개발 조직들은 경험법칙, 선형함수 추정경험 또는 개략적 일정 추정표에 기반하여 개발노력과 개발기간을 추정한다.

경험법칙은 다양한 모델들이 존재하고 있어 특정 프로젝트에 적합한 모델이 어떤 것인지에 대한 기준이 없어 적용에 어려움이 있으며, 선형함수 추정 경험은 기능점수에 기반하고 있으나 너무 단순하여 원하는 정보를 충분히 얻지 못한다. 따라서 최선의 방법은 개략적 일정 추정표에 기반하여 개발노력과 일정을 추정하는 것이 될 수 있다. 개략적 일정 추정표를 적용할 경우 직면하는 문제점은 이 표에서 사용하고 있는 개발기간 용어가 일반적으로 알고 있는 용어와 상이하다는 점이다.

본 논문에서는 개략적 일정 추정표에서 사용하고 있는 개발기간에 대한 용어를 일반적으로 알고 있는 용어와 통일시키는 작업을 수행한 후 개발노력과 개발기간을 추정하는 회귀분석 모델을 제시하였다.

소프트웨어 추정분야에서는 LOC에 비해 FP를 보다 선호하여 소프트웨어 규모를 추정하고 있다. 이에 따라 FP에 기반한 개략적 일정 추정표가 필요하다. 이 경우 최단 개발기간, 효율적 개발기간과 명목상 개발기간에 대한 일정과 노력의 값이 제시되어야 한다. 이는 기존의 선형함수 추정 경험을 확장하거나 벤치마킹 데이터로부터 유도가 필요하다. 따라서 추후 이 분야에 대한 연구를 수행할 예정이다.

참 고 문 헌

[1] Curran, K., "COM 820 - Project Management: Software Project Estimation & Metrics," University of Ulster, Magee College, N. Ireland, 2004.

[2] Jones, C., "Assessment and Control of Software Risks," Englewood Cliffs, N. J., Yourdon Press, 1994.

[3] Charkravarthy, S., "Writing a Software Schedule," The Old Joel on Software Forum, Fog Creek Software, 2003.

[4] Drinka, D., "CIS 410: Project Management," http://gaius.cbpp.uaa.alaska.edu/drinka/cis410/ChapterNotes/Chapter208.doc, 2005.

[5] Thomas, S., "Steven"s Fairly Long and Involved Estimation Procedure," http://www.balagan.org.uk/work/estimate_long.html

[6] Lewis, B., "The 70-Percent Failure," Infoworld, http://archive.infoworld.com/articles/op/xml/01/10/29/ 011029/opservival.html, 2003.

[7] Peters, K., "VB Software Project Estimation," VBPMT Article, http://www.user.dccnet.com/gbsolutionware/References/ProjectMgmt-EstimatingPt2.html.

[8] Mukherji, R., "Business Case For A Common Services Based Architecture," OMG Document #3-300, http://www.omg.org/docs/corbamed/00-03-13.pdf, 2000.

[9] Roetzheim, W., "Estimating Software Costs," Costxpert Group, http://www.costxpert.com/resource_center/ SDarticles/SDarticle1.pdf.

[10] Cukic, B., "WVU LDCSEE CS 430: Project Scheduling and Tracking," http://www.csee.wvu.edu/~cukic/cs430/ ch24.pdf, 2004.

[11] Khaiter, P., "ITEC 4010: System Analysis and Design II," http://www.atkinson.yorku.ca/~pkhaiter/Lectures4010/Lecture4010-13.pdf.

[12] Fakharzadeh, C., "CORADMO in 2001: A RAD Odyssey," 16th International Forum on COCOMO and Software Cost Modeling, USC-CSE, 2001.

[13] Putnam, L. H. and Myers, W., "Familiar Metric Management - Time-to-Market, http://www.qsm.com/fmm_08.pdf

[14] Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R. and Selby, R., "Cost Models for Future Life Cycle Processes: COCOMO 2.0," Annals of Software Engineering, Vol. 1, No. 1, pp. 1-24, 1995.

[15] Carbno, C., "Optimal Resource Allocation for Projects," Project Management Journal, 1999.

[16] Ward, J. A., "Productivity Through Project Management: Controlling the Project Variables," Information Management, 1994.

[17] Boehm, B. W., "Software Engineering Economics," Prentice Hall, 1981.

[18] Pressman, R., Jones, C., Basilli, V. and Putnam, L., "16 Critical Software Practices, Estimate Cost and Schedule Empirically," http://www.iceinusa.com/16CSP/content/2-cost/anmetrgt.html

[19] Holtsinger, M., "CIS 4251/CIS 5930 Software Development," 1999.

[20] Booker, G., "Info 638 Software Project Management: Estimation, WBS, and Scheduling,"2000.

[21] Abts, C., Brown, A. W., Chulani, S., Clark, B. K., Horowitz, E., Madachy, R., Reifer, D. and Steece, B., "Software Cost Estimation with COCOMO II," Prentice-Hall, 2000.

[22] Jones, C., "Applied Software Measurements," McGraw Hill, 1996.

[23] Marin Consultancy, "Estimation, Marin Solutions Technical Paper, 2001.

[24] McConnell, S., "Rapid Development: Taming Wild Software Schedules," Microsoft Press, 1996.

[25] Ardis, M., "Estimation: CSSE 372, Software Project Management," Rose-Hulman Institute, 2004.

[26] Walston, C. E. and Felix, C. P., "A Method of Programming Measurement and Estimation," IBM Systems Journal, Vol. 16, No. 1, pp. 54-73, 1977.

[27] Simmons, D., Ellis, N., Fujihara, H. and Kuo, W., "Software Measurement: A Visualization Toolkit for Project Control and Process Improvement," Prentice Hall, Upper Saddle River, N.J., 1998.

[28] Yang, Y., Chen, Z., Valerdi, R. and Boehm, B., "Effect of Schedule Compression on Project Effort," 5th Joint International Conference and Educational Workshop, Denver, Colorado, 2005.

[29] Dromey, G., "Project Management Health Check," School of CIT, Griffith University, Austoralia, 1997.

[30] Royce, W., "Next-Generation Software Economics," Rational e-zine for the Rational Community, Rational Software, 2000.

[31] Putnam, L. H. and Myers, W., "Familiar Metric Management - The Effort-Time Tradeoff: It"s in the Data," Cutter Consortium"s IT Metrics Strat egies, http://www.qsm.com/fmm_13.pdf

[32] Burris, E., "CS 451: Software Engineering - Measurement and Estimation," School of Interdisplinary Computing and Engineering, University of Missouri, 2002.

[33] Mengny, T., "Concrete Estimation(Size, Effort, Schedule)," Institut Superieur d"Informatique et d"Automatique, http://www.essi.fr/~hugues/GL/Project/estimation.html, 2001.

[34] Richard, L. K., "How Much Is Too Much?," http://www.projectmanager.com/content/regular/art20051128.html, 2005.

[35] Leung H. and Fan, Z., "Software Cost Estimation," Department of Computing," The Hong Kong Polytechnic University.

[36] Johnson, K., "Software Size Estimation," Depart ment of Computer Science, University of Calgary, Alberta, CANADA, http://sern.ucalgary.ca/cources/seng/621/w98/johnsonk/software.html, 1998.

[37] Mildred, L. G., "Practical Software Engineering: Cost and Effort Estimation," University of Calgary, Albert, CANADA, 1996.

[38] Molφkken, K. and Jφrgensen, M., "A Review of Surveys on Software Effort Estimation," IEEE International Symposium on Empirical Software Eng.(ISESE), 2003.

[39] Gruenwalt, H., "Evolving Software Development Tools/Techniques," College of Business, The University of Oklahoma, 1999.

[40] McAulay, K., "Information Systems Development and the Changing Role of IS in the Organization," 1st New Zealand MIS Management Conference, 1987.

[41] Heemstra, F. J. and Kusters, R. J., "Controlling Software Development Costs: A Field Study," International Conference on Organization and Information Systems, 1989.

[42] Wydenbach, G. and Paynter, J., "Software Project Estimation: A Surveys of Practices in New Zealand," New Zealand Journal of Computing, Vol. 1, No. 1B, pp. 317-327, 1995.

[43] Richard, L. K., "Staff IT," http://www.gantthead.com, 2005.

[44] Yang, Y., Chen, Z., Valerdi, R. and Boehm, B., "Effect of Schedule Compression on Project Effort," 5th Joint International Conference & Educational Workshop, Denver, Colorado, 2005.

[45] Putnam, L. H. and Myers, W., "Measures of Excellence: Reliable Software on Time, Within Budget," Englewood Cliffs N.J., Yourdon Press, 1992.

◐ 저 자 소 개 ◑

박 영 목

1990년 경상대학교 전산통계학과 졸업(학사)
1999년 경상대학교 대학원 산업정보공학과 졸업(석사)
2003년 경상대학교 대학원 컴퓨터과학과 졸업(박사)
1996~현재 진주산업대학교 학술정보전산팀장
관심분야 : 소프트웨어공학, 멀티미디어, 원격교육, 신경망 etc.
E-mail : ympark@jinju.ac.kr