韓國컴퓨터情報學會 論文誌
第16卷 第9號, 2011. 9.

2011-16-9-1-5

소프트웨어 비용-일정 타협을 위한 수정된 소프트웨어 공식

이 상 운*, 최 명 복 **

Modified Software Equation for Software Cost-Schedule Tradeoff

Sang-Un, Lee *, Myeong-Bok, Choi **

요 약

하나의 소프트웨어를 개발하기 위해서는 개발조직의 생산성, 투입되는 노력, 개발일정, 소프트웨어 규모 간에 관계가 있다. 이들 관계를 유도한 식이 소프트웨어 공식이다. 소프트웨어 공식은 근본적으로 규모-노력, 규모-일정 관계가 적용되었다. 이 관계가 적절하지 않을 경우 소프트웨어 공식으로부터 유도되는 비용-일정 타협 공식, 투입 인력 프로파일 등의 효용성이 없어진다. 본 논문에서는 이러한 문제점을 해결하고자 수정된 소프트웨어 공식을 유도하였으며, 수정된 소프트웨어 공식에 기반하여 소프트웨어 규모별로 비용-일정을 타협하는 모델들을 제안하였다. 소프트웨어 개발 성공률을 향상시키기 위해 제안된 모델을 적용하면 계약 협상이나 입찰에 도움이 될 것이다.

▸ 키워드 : 소프트웨어 공식, 비용-일정 타협 , 불가능 영역, 공정 생산성, 일정 단축

Abstract

To develop a software, there is a relationship between development organization's productivity, effort, development schedule, and software size . It is software equation that motive these relations. Basically, relationship between size-effort and size-schedule is applied. If this relationship is not proper, there would be no effect of the cost-schedule tradeoff equation that is derived from software equation, and the manpower profile analysis, etc. To solve these unwanted problems, we presented a modified software equation and a cost-schedule tradeoff model based on the modified software equation. To improve software development success rate, applying proposed model will help in contract negotiation or bid.

▸ Keywords : Software Equation, Cost-Schedule Tradeoff, Impossible Region, Process Productivity, Schedule Compression

• 제1저자 : 이상운*, 교신저자 : 최명복**
• 투고일 : 2011. 05. 13, 심사일 : 2011. 06 17, 게재확정일 : 2011. 06. 23.
*, ** 강릉원주대학교 멀티미디어공학과 (Dept. of Multimedia Science, Gangneung-Wonju National University)

I. 서 론

개발하고자 하는 소프트웨어의 규모 (크기)가 추정되었다고 가정할 때, 다음으로 추정해야 하는 것은 이 규모에 기반하여 소프트웨어 개발에 소요되는 일정 (기간)과 노력 (비용)의 양이 된다. 이를 소프트웨어 비용 또는 일정 추정 모델이라 하며, 대표적인 모델이 Boehm의 COCOMO II가 있다.[1] Putnam과 Myers [2-7]는 공정 생산성 (Process Productivity, PP)과 더불어 소프트웨어 규모 (Source Lines of Code, $SLOC$), 노력 (Effort, E), 일정 (Time, T)들 간에 $SLOC = PP \cdot (\frac{E}{B})^{\frac{1}{3}} \cdot T^{\frac{3}{4}}$라는 관계가 성립된다고 하였으며, 이를 소프트웨어 공식 (Software Equation, SE)이라 칭하였다. 또한, 소프트웨어 공식으로부터 $E \times T^4 = K$에 의해 일반적으로 추정된 명목상의 개발일정 (Nominal Time, T_n)에 대해 $75\% T_n \leq T_n \leq 130\% T_n$의 범위에서 소요되는 노력의 양이 결정되며, 이에 따라 비용-일정 타협이 이루어진다. 여기서 제기되는 문제점은 $SLOC = xE^{0.33}$와 $SLOC = yT^{1.33}$ 관계가 성립되지 않으면 소프트웨어 공식으로부터 유도되는 모든 공식들의 효용성이 없어진다. 이러한 문제점을 해결하고자 본 논문에서는 소프트웨어 규모별 명목상 일정과 노력의 기준이 되는 데이터를 획득하였다. 이 데이터에 근거하여 $SLOC = E^a$와 $SLOC = T^b$의 관계를 회귀분석으로부터 유도하여 수정된 소프트웨어 공식 (Modified Software Equation, MSE)을 제안하였다. 다음으로 소프트웨어 공식에 적용된 B 값의 적용성 문제를 다루어 $SLOC = PP \cdot \left(\frac{E}{B}\right)^{\frac{4}{5}} \cdot T^{\frac{12}{5}}$와 $PP \cdot E^{\frac{4}{5}} \cdot T^{\frac{12}{5}}$를 제안하였다. 끝으로, MSE로부터 $E \times T^3 = E(\frac{SLOC}{PP})^{\frac{5}{4}}$ 또는 $(\frac{SLOC}{PP})^{\frac{5}{4}}$를 유도하여 소프트웨어 규모별로 비용-일정을 타협하기 위한 모델들을 제안하였다.

2장에서는 소프트웨어 공식과 비용-일정 타협과 관련된 연구와 문제점을 살펴본다. 3장에서는 먼저, 수정된 소프트웨어 공식을 제안하고 다음으로 B 값의 적용성 여부를 검증한다. 마지막으로 소프트웨어 규모별로 비용-일정을 타협할 수 있는 모델을 제시한다. 기능점수에 기반한 명목상 개발일정 추정 모델을 제시하고, 4장에서는 결론 및 향후 연구 과제를 다룬다.

II. 관련 연구

1. 소프트웨어 공식

Putnam과 Myers[2-7]는 "어떤 생산성 수준에 도달한 개발자는 일정 기간 동안 노력을 투입하면 일정한 신뢰 수준의 산출물을 얻는다."라는 법칙에 근거하여 식 (1)과 같이 5개 척도 (Metrics)들 간의 상호관계를 도출하였다.

Work Product (at a Reliability Level) =

Effort over a Time Interval at a Productivity Level

Size (at Defect Rate) = Effort × Time × Process Productivity

$$Size (at\ Defect\ Rate) = E^a \times T^b \times PP \quad \cdots\cdots (1)$$

여기서, $Size$는 SLOC, E는 년 인원 (Person-Years), T는 년 (Years)의 단위를 적용하고 있다. PP는 어느 한 프로그래머의 생산성이 아닌 개발조직 전체의 생산성이다. 이 용어는 처음에는 상수로 기술 제약사항 (Technology Constraint)으로, 다음에는 전체 조직에 적용하기 위해 조직의 생산성 (Organizational Productivity)으로 변경되었으며, 최근 들어서는 "공정"을 "공정 생산성 (Process Productivity)"으로 사용되고 있다[6].

모수 a와 b는 과거 개발된 소프트웨어 프로젝트들로부터 $SLOC = xE^{0.33}$와 $SLOC = yT^{1.33}$을 유도하여 식 (2)의 소프트웨어 공식 (SE)을 제안하였다[2,8].

$$SLOC = PP \times (\frac{E}{B})^{\frac{1}{3}} \times T^{\frac{4}{3}} \quad \cdots\cdots (2)$$

여기서, B는 숙련도 인자 (Special Skill Factor)로 규모 의존 모수라고도 한다. 이 모수는 작은 규모의 소프트웨어에 보다 큰 가중치를 부여하는 효과를 나타내며 표 1에 제시되어 있다[9,10].

표 1. B 값 적용 기준

Table 1. Applied Criteria of B Value

Size(KLOC)	5-15K	20K	30K	40K	50K	>70K
B	0.16	0.18	0.28	0.34	0.37	0.39

프로젝트의 일정은 순차적인 작업의 제약사항에 영향을 받으며, 임의의 시점에서 최대로 투입할 수 있는 노력 (인원 수)은 독립적인 하위 작업들의 수에 의존하기 때문에 개발일정도 생산성에 영향을 미친다. 그러나 전통적인 생산성은 개발일정을 고려하지 않고 SLOC/E로 계산되어 소프트웨어 개발 생산성 척도로는 적절하지 않다[11]. 따라서 개발조직의 생산성이 노력과 일정 모두에 영향을 받도록 고려한 것이 PP이다. 모수 Size, B, E, T, a, b 값들을 대입하여 $PP = \frac{Size}{(E/B)^{1/3} \times T^{4/3}}$ 식으로 계산된다. 그러나 PP 값만 이용하면 개발조직의 생산성 수준이 어느 정도인지 판단이 불가하여 생산성 지표 (Productivity Index, PI)를 사용하고 있으며, 1 ≤ PI ≤ 40의 범위를 갖고 있다. Putnam과 Myers[2-7]가 제시한 PP와 PI는 표 2에 제시하였다[9,10].

식 (2)로부터 노력 (비용)과 일정을 타협할 수 있는 식 (3)이 유도될 수 있다.

$$E \times T^4 = B\left(\frac{Size}{PP}\right)^3 = K\ (Constant) \quad \cdots\cdots\cdots\cdots (3)$$

표 2. PP와 PI 값 변환 기준
Table 2. PP and PI Value Transform Criteria

PI	PP	시스템명	PI	PP	시스템명
		-			
1	754	마이크로코드	21	92,736	-
2	987	-	22	121,393	-
3	1,220	펌웨어 (ROM)	23	150,050	-
4	1,597	실시간 내장형, 항공전자	24	196,418	-
5	1,974	-	25	242,786	-
6	2,584	레이더 시스템	26	317,811	-
7	3,194	명령 & 통제	27	392,836	-
8	4,181	공정관리	28	514,229	-
9	5,186	-	29	635,622	-
10	6,765	통신	30	832,040	-
11	8,362	-	31	1,028,458	-
12	10,946	시스템소프트웨어,	32	1,346,269	-
13	13,530	과학시스템	33	1,664,080	-
14	17,711	-	34	2,178,309	-
15	21,892	-	35	2,692,538	-
16	28,657	비즈니스 시스템	36	3,524,578	-
17	35,422	-	37	-	-
18	46,368	-	38	-	-
19	57,314	-	39	-	-
20	75,025	-	40	-	-

식 (3)에 따라 개발조직의 생산성 (PP)과 개발할 소프트웨어의 규모 (SLOC)만 알고 있다면 개발에 투입할 노력 (E)과 일정 (T)간에 타협이 가능하다. 또한, 인력을 점진적으로 증가시키는 곡선의 기울기 MBP, (Manpower Buildup Parameter) = K/T^3와 MBI, (Manpower Buildup Index) = E/T^3를 적용하면 개발인력의 프로파일을 구할 수 있다.

2. 소프트웨어 공식의 문제점

첫 번째로, 식 (3)에 의하면 임의의 규모를 가진 소프트웨어를 개발하는데 있어 개발일정은 투입되는 노력의 4승의 가중치로 소프트웨어 규모에 영향을 미친다. 즉, 소프트웨어를 성공적으로 개발하기 위해서는 많은 양의 노력을 증가 또는 감소시키는 것 보다 적은 양의 개발일정을 단축 또는 연장시키는 것이 보다 큰 영향을 미친다. 따라서 적절한 개발일정 설정이 프로젝트의 성공 여부를 결정할 수 있는 요인이 될 수 있다. 소프트웨어 개발시 적용되는 모든 공식의 근간이 되는 것은 $Size = xE^{0.33}$와 $Size = yT^{1.33}$의 관계이다. 이들 관계가 성립되지 않을 경우 식 (2)의 SE로부터 유도되는 모든 식들은 효용성이 없어지며, 임의의 소프트웨어를 신규로 개발할 경우 적용할 개발노력과 개발일정 간에 타협이 불가하다. 따라서 $Size = PP \cdot E^a \cdot T^b = PP \cdot (E/B)^{0.33} \cdot T^{1.33}$에서, a와 b 값이 적절한 값인지 판단할 필요성이 제기된다.

두 번째로, E 값의 해석 및 적용 문제이다. 표 1에서 주어진 규모를 범위의 최대값으로 해석 하였을 경우, 5 ~ 15K = 0.16, 15.01 ~ 20K = 0.18, 20.01 ~ 30K = 0.28, 30.01 ~ 40K = 0.34, 40.01 ~ 50K = 0.37, 70K 이상 = 0.39로 적용할 수 있다. 여기서, 0.01 ~ 5K, 50.01 ~ 60K, 60.01 ~ 70K 범위에 속하는 소프트웨어는 어떤 값을 적용해야 하는지가 문제로 제기된다. 또한, 70K ~ 수백 K에 속하는 광범위한 규모의 소프트웨어가 모두 동일한 0.39로 적용해도 문제가 없는지에 대한 타당한 근거가 제시되어 있지 않다.

본 논문에서는 먼저, $Size = xE^{0.33}$와 $Size = yT^{1.33}$ 관계가 있는지 검증하고 적절한 SE를 유도하고자 하며, 이를 "수정된 소프트웨어 공식 (MSE)"이라 칭한다. 다음으로 E 값의 적용 여부를 검증하였다. 마지막으로 제안된 MSE에 기반하여 비용-일정 타협 모델을 제시한다.

3. 비용-일정 타협 관련연구 및 문제점

하나의 소프트웨어를 개발할 경우 명목상으로 필요한 일정을 T_n, 개발일정을 최대한 단축시킬 수 있는 시점을 T_c, 개발일정을 최대한 연장시킬 수 있는 한계를 T_s라 하자.

소프트웨어 개발 분야에서는 명목상 개발일정을 단축시키

기 위해 개발인력을 무한히 추가하더라도 어느 시점 이하에서는 시스템을 성공적으로 완료할 수 없는 최소한의 개발기간이 존재하며, 이를 불가능 영역 (Impossible Region) 이라 부른다[3,11-14]. 프로젝트를 성공적으로 완료하기 위해서는 불가능 영역을 회피할 수 있어야만 한다. 그러나 고객은 개발 불가능 일정에 상관없이 보다 빠르게 제품을 획득하려는 경향이 있다. 반면에 개발자는 단기간에 개발하는데 동의하지 않을 것이다. 왜냐하면 단기간에 개발할 경우 소요되는 노력이 크게 증가하며, 이로 인한 간접비용과 결함도 추가로 증가하기 때문이다. 따라서 개발노력과 개발일정 모두를 고려하여 적절한 값을 설정하는 것이 중요하다. 이를 비용-일정 타협 (Cost-Schedule Tradeoff)이라 하며, 하나의 소프트웨어를 개발하기 위한 일정과 노력 (비용) 간에는 그림 1의 관계를 갖고 있다.

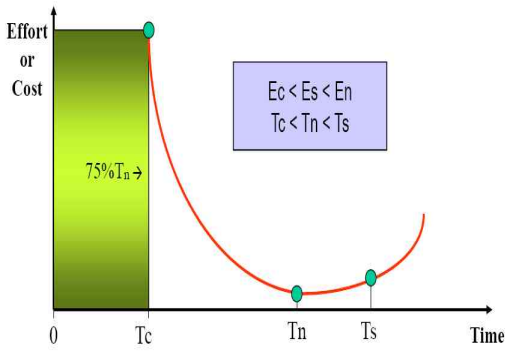


그림 1. 비용-일정 타협
Fig. 1. Tradeoff Cost and Schedule

실제 프로젝트들을 대상으로 한 경험적 결과로 얻은 T_c와 T_s 연구 결과는 표 3과 같이 다양하게 제시되어 있다. 이와 같이 다양하게 정의된 T_c와 T_s 값들로부터 실제로 어느 기준을 적용할 것인가에 혼란을 초래하고 있다.

그러나 T_c에 대해 대부분의 연구자들은 T_n의 약 75%에 한계가 있다고 동의하고 있다($0.75\%T_n \le T_c \le 0.80\%T_n$) [1,14-16]. 또한, T_s는 일반적으로 $130\%T_n$으로 제시하고 있는 반면에 Putnam과 Myers[3]는 T_s는 $130\%T_c$를 제시하고 있다.

결론적으로 일반적 결과인 $T_c = 75\%T_n$, $T_s = 130\%T_n$을 적용하는 것이 타당해 보이며, 본 논문에서는 이 기준을 적용하여 비용-일정 타협 모델을 제시한다.

III. 수정된 소프트웨어 공식

1. 실험에 적용된 데이터

표 3. T_c와 T_s의 정의
Table 3. Definition of T_c and T_s

참고문헌		T_c	T_s	분석 대상
Sifri[17]		$75\%T_n$	$2T_n$	750개 프로젝트
Carbno[13], Ward[18]		-	1명 개발기간	Rules of Thumb
COCOMO II 2000[1]		$75\%T_n$	$130\%T_n$	161개 프로젝트
Putnam과 Myers[2,5]		-	$130\%T_c$	4000개 프로젝트
Construx[19]		$75\%T_n$	-	Rules of Thumb
Simons [20]	PRICE	$70\%T_n$	$130\%T_n$	-
	DSN	$70\%T_n$	$130\%T_n$	-
	COCOMO	$75\%T_n$	$130\%T_n$	-
	Jenson	$85\%T_n$	$120\%T_n$	-

MSE을 유도하기 위해 적용된 데이터는 Menguy[21]로부터 획득하였으며, 이 데이터는 Putnam과 Myers[4], Boehm[14], Kemerer[22], Jones[16,23], McConnell[24]로부터 유도되어 대체로 타당한 (어림짐작) 추정 (Ballpark Estimation) 값으로 제안되었다. Menguy[21]가 제시한 데이터는 시스템 소프트웨어, 비즈니스 소프트웨어와 수축포장된 (Shrink-wrap) 상용 패키지 소프트웨어로 분류하여 제시하고 있다. 시스템 소프트웨어는 운영체제, 장치 구동기, 컴파일러, 코드 라이브러리 등의 소프트웨어에 관한 사항으로 최근에 중요하게 대두되고 있는 내장형, 펌웨어, 실시간 시스템과 과학용 소프트웨어는 포함되어 있지 않은 문제점이 있다. 비즈니스 소프트웨어는 단일 조직에 의해 사용되는 회사 내 시스템으로 인사관리 시스템, 회계관리 시스템, 재고관리 시스템, 정보시스템 (IS), 정보기술 (IT) 시스템과 관리정보 시스템 (MIS)을 포함하고 있다. 상용 패키지 소프트웨어는 패키지화하여 상용으로 판매되는 워드프로세서, 스프레드시트, 회계분석 시스템 등으로 구성되어 있다.

본 논문에서는 개발의 대부분을 차지하고 있는 비즈니스 소프트웨어를 대상으로 소프트웨어 공식을 유도하고자 한다. 비즈니스 소프트웨어에 대한 소프트웨어 규모별 개발노력과 개발일정에 대한 자료는 표 4에 제시되어 있다.

여기서 최단 개발 (Shortest Development, D_s)은 이론상으

로는 달성이 불가능한 개발기간을, 효율적 개발 (Efficient Development, D_e)은 능력 있는 팀과 훌륭한 관리 하에서 달성할 수 있는 개발기간이다. 또한 명목상 개발 (Nominal Development, D_n)은 보통의 팀을 구성하였을 때 일반적으로 프로젝트들이 달성할 수 있는 개발기간이다. Menguy[21]가 제시한 데이터에는 월 인원과 월이 단위로 사용되었다. 우리는 MSE를 제안하기 위해 SE에서 제안한 PP 값을 얻으려면 식 (2)의 SE에서 E와 T에 적용된 년 인원과 년 단위와 일치시켜야 한다. 따라서 원 인원을 년 인원으로, 월을 년으로 변환시킨 값도 함께 제시하였다.

표 4. 비즈니스 프로젝트의 규모별 개발노력과 일정
Table 4. Effort and Schedule of Business Project based on Size

규모 (SLOC)	최단 개발 (D_s)		효율적 개발 (D_e)		명목상 개발 (D_n)	
	노력 월(년)인원	일정 월(년)	노력 월(년)인원	일정 (월/년)	노력 월(년)인원	일정 (월/년)
10,000	5(0.42)	3.5(0.29)	5(0.42)	4.9(0.41)	9(0.75)	6(0.50)
15,000	8(0.67)	4.1(0.34)	8(0.67)	5.8(0.48)	15(1.25)	7(0.58)
20,000	11(0.92)	4.6(0.38)	11(0.92)	7(0.58)	21(1.75)	8(0.67)
25,000	15(1.25)	5.1(0.43)	14(1.17)	7(0.58)	27(2.25)	9(0.75)
30,000	22(1.83)	5.5(0.46)	20(1.67)	8(0.67)	37(3.08)	9(0.75)
35,000	26(2.17)	5.8(0.48)	24(2.00)	8(0.67)	44(3.67)	10(0.83)
40,000	34(2.83)	6(0.50)	30(2.50)	9(0.75)	54(4.50)	10(0.83)
45,000	39(3.25)	6(0.50)	34(2.83)	9(0.75)	61(5.08)	11(0.92)
50,000	46(3.83)	7(0.58)	40(3.33)	10(0.83)	71(5.92)	11(0.92)
60,000	57(4.75)	7(0.58)	49(4.08)	10(0.83)	88(7.33)	12(1.00)
70,000	71(5.92)	8(0.67)	61(5.08)	11(0.92)	105(8.75)	13(1.08)
80,000	83(6.92)	8(0.67)	71(5.92)	12(1.00)	125(10.42)	14(1.17)
90,000	96(8.00)	9(0.75)	82(6.83)	12(1.00)	140(11.67)	15(1.25)
100,000	110(9.17)	9(0.75)	93(7.75)	13(1.08)	160(13.33)	15(1.25)
120,000	140(11.67)	10(0.83)	115(9.58)	14(1.17)	200(16.67)	16(1.33)
140,000	160(13.33)	10(0.83)	140(11.67)	15(1.25)	240(20.00)	17(1.42)
160,000	190(15.83)	10(0.83)	160(13.33)	15(1.25)	280(23.33)	18(1.50)
180,000	220(18.33)	11(0.92)	190(15.83)	16(1.33)	330(27.50)	19(1.58)
200,000	250(20.83)	11(0.92)	210(17.50)	17(1.42)	370(30.83)	20(1.67)
250,000	330(27.50)	13(1.08)	280(23.33)	19(1.58)	480(40.00)	22(1.83)
300,000	420(35.00)	14(1.17)	345(28.75)	20(1.67)	600(50.00)	24(2.00)
400,000	590(49.17)	15(1.25)	490(40.83)	22(1.83)	840(70.00)	27(2.25)
500,000	780(65.00)	17(1.42)	640(53.33)	25(2.08)	1,100(91.67)	29(2.42)

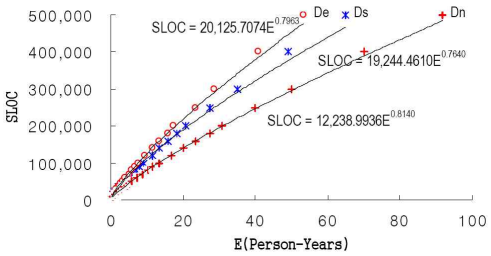
2.. 수정된 소프트웨어 공식 유도

표 4를 대상으로 SLOC vs. E, SLOC vs. T 관계로부터 $SLOC = xE^a$와 $SLOC = yT^b$ 회귀분석을 이용하여 그림 2의 모델을 얻었으며, 모델의 성능은 표 5와 같다. 여기서, *는 D_s (최단 개발), o는 D_e (효율적 개발), +는 D_n (명목상 개발)을 나타내고 있다. 표에서 모델들의 결정계수가 대부분 99% 이상으로 주어진 데이터를 잘 표현하는 모델임을 알 수 있다.

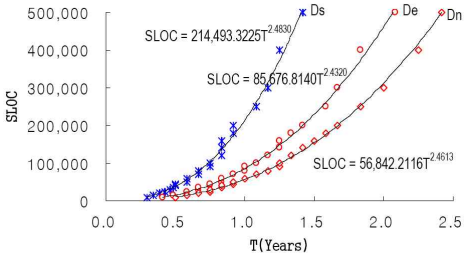
표 5로부터 D_s는 $SLOC = PP \cdot (E/B)^{0.76} \cdot T^{2.48}$, D_e는 $SLOC = PP \cdot (E/B)^{0.80} \cdot T^{2.43}$, D_n은 $SLOC$ $PP \cdot (E/B)^{0.81} \cdot T^{2.46}$를 얻을 수 있다. 결국, $Size \propto E^{1/3} \cdot T^{4/3}$ 의 관계가 성립되지 않아 SE를 적용할 수 없음을 알 수 있다.

식 (2) $Size = PP \cdot (E/B)^{\alpha} \cdot T$에서 $a:b = 0.33:1.33 = 1:4$로 일정이 노력에 비해 4승의 영향을 미친다. 반면에 D_s의 $a:b = 0.76:2.48 = 1:3.25 \simeq 1:3$, D_e의 $a:b = 0.80:2.43 = 1:3.04 \simeq 1:3$, D_n의 $a:b = 0.81:2.41 = 1:2.98 \simeq 1:3$이 된다. 즉, 개발될 소프트웨어 규모에 대해 개발일정은 개발노력의 3 승의 영향을 미친다.

현실적으로 우리가 달성 가능한 개발일정은 D_s 보다는 D_e 또는 D_n일 것이다. D_e과 D_n의 모수 관계인 $a:b \simeq 0.8:2.4$로부터, MSE는 식 (4)로 정의될 수 있다. 즉, 명목상 개발과 효율적 개발의 경우 적용되는 소프트웨어 공식은 동일하며, 단지, 적용되는 PP 값에 차이가 있을 뿐이다. 식 (4)를 B 값을 고려한 MSE (MSE with B)라 하자.



(a) SLOC vs. E



(b) SLOC vs. T

그림 2. 소프트웨어 규모별 개발노력과 개발일정 회귀모델
Fig. 2. Effort and Schedule Regression Model based on Size

MSE with B : $SLOC = PP \cdot (\frac{E}{B})^{\frac{4}{5}} \cdot T^{\frac{12}{5}}$ ..... (4)

식 (2)의 SE와 식 (4)의 MSE에 따라 얻은 PP의 값은 표 6

에 제시하였다.

일반적으로 소프트웨어 규모가 커지면 생산성은 감소하는 경향을 나타낸다. 그러나 SE의 경우 소프트웨어 규모가 증가하면 PP 또는 PI 값도 함께 증가하는 경향을 나타내는데 반해 MSE는 소프트웨어 규모가 증가하면 PP 또는 PI 값은 감소하는 경향을 나타내고 있어 일반적인 소프트웨어 규모와 생산성 관계를 적절히 반영하고 있다. 따라서 MSE가 SE 보다 PI 값을 계산하는데 보다 유용함을 알 수 있다.

표 5. 규모와 개발노력, 개발일정 관계식
Table 5. Size, Effort, and Schedule Formulas

구분	Size vs. E 관계식 (결정계수)	Size vs. T 관계식 (결정계수)
D_s	$SLOC = 19,244.4610E^{0.7}$ ($R^2 = 0.9975$)	$SLOC = 214,493.3225T^{2.4830}$ ($R^2 = 0.9935$)
D_e	$SLOC = 20,125.7074E^{0.79}$ ($R^2 = 0.9988$)	$SLOC = 85,676.8140T^{2.4320}$ (($R^2 = 0.9962$)
D_n	$SLOC = 12,238.9936E^{0.8}$ ($R^2 = 0.9994$)	$SLOC = 56,842.2116T^{2.4613}$ (($R^2 = 0.9975$)

MSE로 얻은 PI 값들을 분석하여 보자. D_n에서 D_e로, D_e에서 D_s로 전환하기 위해 개발조직이 향상시켜야 할 생산성은 전반적으로 볼 때, D_n에서 D_e로는 PI가 3~4 (평균 3.5), D_e에서 D_s로 전환시키는데 3~4 (평균 3.2)가 향상되어야 한다는 법칙을 발견할 수 있다.

표 6. E를 고려한 PP와 PI 계산
Table 6. Compute PP and PI with E

규모 (SLOC)	PP (PI)					
	SE with E			MSE with E		
	D_s	D_e	D_n	D_s	D_e	D_n
10,000	27,162(16)	17,362(14)	10,924(12)	40,828(18)	18,208(15)	6,998(11)
15,000	28,268(16)	17,821(15)	11,278(13)	28,763(17)	12,511(13)	4,818(9)
20,000	41,839(18)	23,937(16)	16,191(14)	54,312(19)	19,828(15)	8,579(12)
25,000	47,617(19)	31,970(17)	18,427(15)	58,885(20)	29,101(17)	9,414(12)
30,000	45,545(18)	28,555(16)	19,929(15)	43,393(18)	19,054(15)	8,780(12)
35,000	49,957(19)	33,444(17)	20,360(15)	45,543(18)	22,441(16)	8,089(11)
40,000	49,953(19)	30,360(17)	21,737(15)	38,715(18)	16,171(14)	7,847(11)
45,000	55,229(19)	33,700(17)	21,279(15)	41,758(19)	17,611(14)	6,816(11)
50,000	47,340(19)	30,849(17)	22,488(16)	28,085(16)	13,343(13)	6,708(10)
60,000	53,395(19)	34,927(17)	22,591(16)	29,002(17)	13,906(14)	5,620(10)
70,000	48,511(19)	33,393(17)	22,353(16)	20,601(15)	10,832(12)	4,698(9)
80,000	53,110(19)	32,610(17)	22,042(16)	21,215(15)	9,084(12)	3,991(8)
90,000	48,690(19)	34,984(17)	21,793(15)	16,013(14)	9,107(12)	3,475(8)
100,000	51,724(19)	33,523(17)	23,170(16)	15,956(14)	7,551(11)	3,470(8)
120,000	49,825(19)	33,985(17)	23,706(16)	12,261(13)	6,400(10)	2,983(7)
140,000	55,624(19)	33,900(17)	24,024(16)	12,855(13)	5,406(10)	2,601(7)
160,000	60,065(20)	37,072(18)	24,185(16)	12,804(13)	5,552(10)	2,291(6)
180,000	56,717(19)	36,166(18)	23,984(16)	10,191(12)	4,662(9)	1,985(6)
200,000	60,416(20)	35,867(18)	23,969(16)	10,223(12)	4,134(8)	1,779(5)
250,000	55,180(19)	35,166(17)	24,221(16)	6,853(11)	3,144(7)	1,437(4)
300,000	55,410(19)	36,793(18)	24,052(16)	5,676(10)	2,822(7)	1,170(3)
400,000	60,252(20)	38,492(18)	24,537(16)	4,886(9)	2,261(6)	899(2)
500,000	58,153(20)	37,167(18)	25,516(16)	3,618(8)	1,680(5)	763(2)

다음으로, E 값을 고려하지 않은 경우를 살펴보자. 이 경우 SE는 식 (5)로, MSE는 식 (6)으로 표현된다. 식 (5)와 (6)으로 구한 PP와 PI 값은 표 7과 같다.

SE without E : $SLOC = PP \times E^{\frac{1}{3}} \times T^{\frac{4}{3}}$ ..... (5)

MSE without E : $SLOC = PP \times E^{\frac{4}{5}} \times T^{\frac{12}{5}}$ ..... (6)

표 7. E를 고려하지 않은 경우의 PP와 PI
Table 7. Compute PP and PI Without E

규모 (SLOC)	PP (PI)					
	SE without E			MSE without E		
	D_s	D_e	D_n	D_s	D_e	D_n
10,000	68,734(20)	43,936(18)	27,644(16)	387,651(27)	172,439(24)	66,439(20)
15,000	71,534(20)	45,098(18)	28,539(16)	273,098(26)	118,788(22)	45,747(18)
20,000	73,679(20)	42,153(18)	28,512(16)	214,130(25)	78,174(21)	33,824(17)
25,000	72,477(20)	48,661(19)	28,047(16)	163,034(24)	80,572(21)	26,064(16)
30,000	69,323(20)	43,462(18)	30,333(17)	120,141(22)	52,755(19)	24,308(16)
35,000	71,320(20)	47,745(19)	29,052(17)	107,955(21)	53,194(19)	19,173(15)
40,000	71,313(20)	43,342(18)	31,032(17)	91,769(21)	38,332(18)	18,601(15)
45,000	76,676(21)	46,787(19)	29,542(17)	92,508(21)	39,015(18)	15,100(14)
50,000	65,723(20)	42,828(18)	31,221(17)	62,217(20)	29,560(17)	14,859(14)
60,000	73,481(20)	48,065(19)	31,089(17)	62,892(20)	30,156(17)	12,187(13)
70,000	66,760(20)	45,954(18)	30,761(17)	44,674(18)	23,489(16)	10,187(13)
80,000	72,465(20)	44,494(18)	30,075(17)	45,060(18)	19,294(15)	8,477(12)
90,000	66,434(20)	47,732(19)	29,734(17)	34,011(17)	19,344(15)	7,381(11)
100,000	70,573(20)	45,740(18)	31,614(17)	33,891(17)	16,037(14)	7,370(11)
120,000	67,983(20)	46,370(19)	32,345(17)	26,041(16)	13,592(14)	6,336(10)
140,000	75,894(21)	46,254(18)	32,779(17)	27,303(16)	11,481(13)	5,524(10)
160,000	81,954(21)	50,582(19)	32,998(17)	27,196(16)	11,792(13)	4,865(9)
180,000	77,386(21)	49,345(19)	32,724(17)	21,646(15)	9,903(12)	4,215(9)
200,000	82,432(21)	48,937(19)	32,704(17)	21,713(15)	8,781(12)	3,779(8)
250,000	75,288(21)	47,982(19)	33,048(17)	14,556(14)	6,677(10)	3,051(7)
300,000	75,603(21)	50,201(19)	32,816(17)	12,056(13)	5,995(10)	2,486(6)
400,000	82,209(21)	52,519(19)	33,479(17)	10,379(12)	4,803(9)	1,909(5)
500,000	79,346(21)	50,712(19)	34,815(17)	7,684(11)	3,567(8)	1,620(5)

표 6과 표 7을 비교시 E 값 여부에 상관없이 SE는 거의 일정한 상수 값을 나타내고 있는 반면 MSE는 소프트웨어 규모가 증가하면 PI 값은 감소하는 경향을 알 수 있다. 표 6과 표 7의 소프트웨어 규모 변화에 따른 PI 값의 범위를 비교한 결과는 표 8에 제시하였다.

SE의 경우 E 값을 고려한 경우 PI 값 범위는 5, E 값을 고려하지 않은 경우 PI는 2의 범위만을 나타내고 있다. 반면에 MSE의 경우 E 값을 고려한 경우 PI 값 범위는 10에서 11을, E 값을 고려하지 않은 경우 PI는 16에서 17의 범위를 나타내고 있다. 따라서 소프트웨어 공식 유도에 있어 E값이 거의 영향을 미치지 않는다. 그러므로 식 (4) 보다는 E 값을 고려하지 않은 간략화한 식 (6)을 적용해도 될 것이다.

표 8. PI 값 변화
Table 8. Variation of PI Value

구분		PI 값 변화 (범위)		
		D_s	D_e	D_n
SE	with E	16 ~ 20(5)	14 ~ 18 (5)	12 ~ 16 (5)
	without E	20 ~ 21 (2)	18~ 19 (2)	16 ~ 17 (2)
MSE	with E	9 ~ 18 (10)	5 ~ 15 (11)	1 ~ 11 (11)
	without E	11 ~ 27 (17)	8 ~ 24 (17)	5 ~ 20 (16)

3. 비용-일정 타협

그림 1과 표 4로부터 개발될 소프트웨어의 T_c와 T_s를 적용하여 비용-일정 타협을 하기 위해서는 효율적 개발 (D_e)과 명목상 개발 (D_n)에 대한 명목상 일정(T_n)과 이 일정에 소요되는 노력(E_n)의 값을 정확히 아는 것이 필요하다. 그러나 기존 연구 대부분은 T_n과 E_n에 대한 근거자료를 제시하지 않고 있어 비용-일정 타협을 할 수 없었다. 반면에 Menguy[21]가 제시한 데이터는 명목상 개발과 효율적 개발에 대한 T_n과 E_n의 값에 대한 개발기간과 개발노력 기준을 제공하고 있으므로, 이를 근거로 비용-일정 타협이 가능하다.

식 (4)와 식 (6)으로 부터 개발노력 E는 식 (7)로 구해지며, 식 (8)에 따라 비용-일정 타협이 이루어진다.

$$E = \frac{B\left(\frac{SLOC}{PP}\right)^{\frac{5}{4}}}{T^3} \quad \text{또는} \quad E = \frac{\left(\frac{SLOC}{PP}\right)^{\frac{5}{4}}}{T^3} \quad \cdots\cdots\cdots\cdots (7)$$

$$B = T^3 \left(\frac{SLOC}{PP}\right)^{\frac{5}{4}} \text{ 또는 } B = T^3 \left(\frac{SLOC}{PP}\right)^{\frac{5}{4}} \quad \cdots\cdots (8)$$

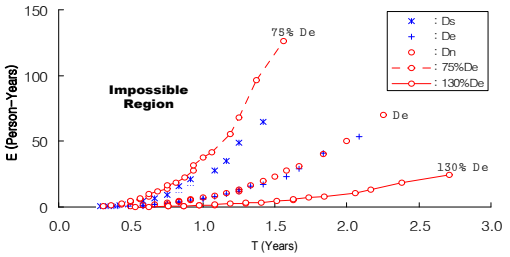
식 (8)을 적용하여 개발일정 단축 한계인 $T_c = 75\% T_n$와 개발일정 연장 한계인 $T_s = 130\% T_n$에 대한 개발노력을 계산할 수 있다. 표 4의 D_e와 D_n의 E와 T의 값에 대한 $75\% D_e$와 $130\% D_e$, $75\% D_n$과 $130\% D_n$은 그림 3과 같다.

T_e와 T_n 각각에 대해 70% ~ 130%까지 5% 간격으로 T의 값에 따른 E의 값 관계를 그린 결과는 그림 4에, 비용-일정 타협 모델은 표 9에 제시하였다.

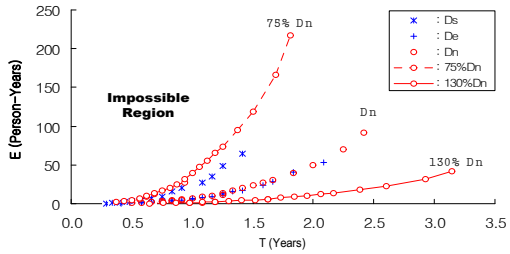
그림 4는 마치 소뿔 (Horn)과 같은 형상을 나타내고 있다. 따라서 이를 "소뿔 비용-일정 타협 모델"이라 칭하자. 신규로 개발될 소프트웨어의 규모만 타당성 있게 추정하였다면 표 9의 해당 규모에 대한 모델을 적용하면 비용-일정 타협을 할 수 있어 고객과의 계약 협상이나 입찰시 활용이 가능할 것이다.

IV. 결 론

소프트웨어를 신규로 개발하고자 할 때, 소요되는 노력과 개발기간에 대한 정확한 정보를 알 수 있느냐 여부가 프로젝트의 성공 여부를 결정한다고 할 수 있다. 이에 대해 Putnam과 Myers[2-7]는 소프트웨어 공식을 제안하였다. 소프트웨어 공식은 소프트웨어 규모 (SLOC)는 노력(E)의 0.33승과 개발기간(T)의 1.33승에 비례한다는 관계로부터 $SLOC = PP$



(a) D_e의 비용-일정 타협 범위



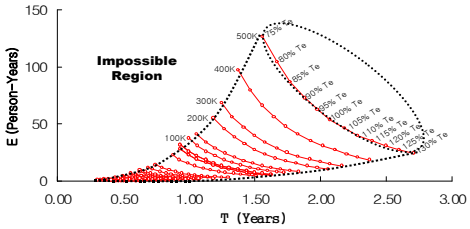
(b) D_n의 비용-일정 타협 범위

그림 3. 비용-일정 타협 범위
Fig. 3. Tradeoff Range of Cost-Schedule

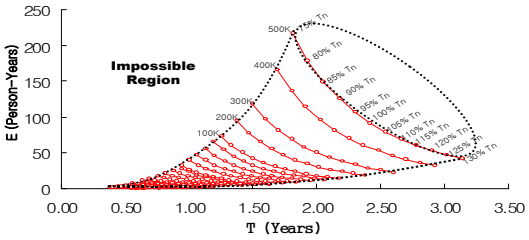
표 9. 비용-일정 타협 모델
Table 9. Cost-Schedule Tradeoff Model

SLOC	D_e			D_n		
	T_e	E_e	비용-일정 타협 모델	T_n	E_n	비용-일정 타협 모델
10,000	0.41	0.62	$E = 0.0263\,T^{-3.0366}$	0.50	0.75	$E = 0.0922\,T^{-3.0417}$
15,000	0.48	0.67	$E = 0.0752\,T^{-3.0071}$	0.58	1.25	$E = 0.2433\,T^{-3.0424}$
20,000	0.58	0.92	$E = 0.1791\,T^{-3.0366}$	0.67	1.75	$E = 0.5211\,T^{-2.9872}$
25,000	0.58	1.17	$E = 0.7961\,T^{-0.6876}$	0.75	2.25	$E = 0.9566\,T^{-2.9840}$
30,000	0.67	1.67	$E = 0.4951\,T^{-2.9924}$	0.75	3.08	$E = 1.3125\,T^{-2.9826}$
35,000	0.67	2.00	$E = 0.5947\,T^{-2.9905}$	0.83	3.67	$E = 2.1181\,T^{-3.0340}$
40,000	0.75	2.50	$E = 1.0657\,T^{-2.9794}$	0.83	4.50	$E = 2.5981\,T^{-3.0360}$
45,000	0.75	2.83	$E = 1.2067\,T^{-2.9807}$	0.92	5.08	$E = 3.9173\,T^{-3.0094}$
50,000	0.83	3.33	$E = 1.9260\,T^{-3.0324}$	0.92	5.92	$E = 4.5614\,T^{-3.0078}$
60,000	0.83	4.08	$E = 2.3575\,T^{-3.0355}$	1.00	7.33	$E = 7.3310\,T^{-3.0006}$

70,000	0.92	5.08	$E=3.9173T^{-3.0094}$	1.08	8.75	$E=11.1405T^{-3.0038}$
80,000	1.00	5.92	$E=5.9163T^{-3.0009}$	1.17	10.42	$E=16.5787T^{-3.0007}$
90,000	1.00	6.83	$E=6.8317T^{-3.0012}$	1.25	11.67	$E=22.8585T^{-3.0010}$
100,000	1.08	7.75	$E=9.8676T^{-3.0023}$	1.25	13.33	$E=26.1187T^{-2.9995}$
120,000	1.17	9.58	$E=15.2516T^{-3.0001}$	1.33	16.67	$E=39.5553T^{-3.0035}$
140,000	1.25	11.67	$E=22.8595T^{-3.0010}$	1.42	20.00	$E=56.6864T^{-2.9886}$
160,000	1.25	13.33	$E=26.1187T^{-2.9995}$	1.50	23.33	$E=79.4642T^{-3.0094}$
180,000	1.33	15.83	$E=37.5738T^{-3.0032}$	1.58	27.50	$E=109.1164T^{-2.9859}$
200,000	1.42	17.50	$E=49.5914T^{-2.9887}$	1.67	30.83	$E=142.4760T^{-2.9966}$
250,000	1.58	23.33	$E=93.4119T^{-3.0157}$	1.83	40.00	$E=249.3159T^{-3.0157}$
300,000	1.67	28.75	$E=132.8175T^{-2.9960}$	2.00	50.00	$E=399.8385T^{-2.9992}$
400,000	1.83	40.83	$E=254.4977T^{-3.0155}$	2.25	70.00	$E=798.9630T^{-3.0005}$
500,000	2.08	53.33	$E=483.0657T^{-3.0014}$	2.42	91.67	$E=1287.2209T^{-2.9939}$



(a) D_e의 비용-일정 타협



(b) D_n의 비용-일정 타협

그림 4. 비용-일정 타협
Fig. 4. Tradeoff Cost-Schedule

• $(E/B)^{0.33}$ • $T^{1.33}$ 을 유도하였으며, $E \times T^4 = K$에 의해 비용-일정 타협을 하였다.

본 논문에서는 소프트웨어 공식의 근간이 되는 $SLOC \propto E^{0.33}$과 $SLOC T^{1.33}$ 관계가 있는지 고찰하여 수정된 소프트웨어 공식을 제안하였다. 제안된 MSE는 $SLOC \propto E^{0.80}$과 $SLOC \propto T^{2.40}$의 관계식 유도로부터 $SLOC = PP \bullet (E/B)^{4/5} \bullet T^{12/5}$를 제안하였으며 이를 B 의 값을 고려한 (with E) 수정된 소프트웨어 공식이라 칭하였다. B 값은 소프트웨어 공식에 큰 영향을 미치지 않음을 증명하여 E를 고려하지 않은 (without E) 수정된 소프트웨어 공식인 $SLOC = PP \bullet E^{4/5} \bullet T^{12/5} E^{4/5}$도 제안하였다. 마지막으로 $E \bullet T^3 = K$에 의해 소프트웨어 규모별 비용-일정 타협을 수행할 수 있는 모델들을 제시하였다.

본 제안된 모델은 소프트웨어 규모를 SLOC를 적용하였다. 그러나 SLOC의 측정 방법의 다양성과 동일한 기능에 대해 구현되는 개발언어별 SLOC의 차이 등으로 인해 소프트웨어 규모로서의 SLOC의 효용성은 좋지 못한 상태이다. 국내에서도 이의 대안으로 기능점수 (Function Point, FP)에 기반하여 소프트웨어 비용을 산출하고 있다. 따라서 추후 ISBSG Benchmark[25] 데이터를 활용하여 기능점수 기반 소프트웨어 공식과 비용-일정 타협 모델의 연구를 수행할 예정이다.

참고문헌

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

[2] L. H. Putnam and W. Myers, "Five Core Metrics: The Intelligence Behind Successful Software Management", Dorset House Publishing, 2003.

[3] L. H. Putnam and W. Myers, "Familiar Metric Management - Time-to-Market", Familiar Metric Management Series, Quantitative Software Management Inc., 2011.

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

[5] L. H. Putnam and W. Myers, "What We Have Learned", The Journal of Defense Software Engineering, 2000.

[6] R. S. Pressman, "Software Engineering-A Practitioner's Approach (6th Ed.)", pp. 679, McGrew-Hill, 2005.

[7] L. H. Putnam and W. Myers, "Familiar Metric Management +", Familiar Metric Management Series, Quantitative Software Management Inc., 2011.

[8] J. Nyman, "Metrics and Measures", GlobalTester, TechQA, 2005.

[9] S. Callaghan, "Index Based Productivity Bench marking", http://www.qsma.com/docs/QSM_A_Productivity_Benchmarking.pdf, QSM Associates, Inc, 2011.

[10] T. McGilbon, “Modern Empirical Cost & Schedule Estimation Tools”, DoD DACS Technical Reports, 1997.
[11] F. P. Brooks, “The Mythical Man-Month: Essays on Software Engineering”, Addison-Wesley, 1995.
[12] D. R. Jones, “Project Scheduling”, Augsburg College, 1999.
[13] C. Carbno, “Optimal Resource Allocation for Projects”, Project Management Journal, 1999.
[14] B. W. Boehm, “Software Engineering Economics”, Prentice Hall, 1981.
[15] Y. Yang, Z. Chen, R. Valerd, and B. Boehm, “Effect of Schedule Compression on Project Effort”, 27th Annual Conference of the International Society of Parametric Analysis, Denver, Colorado, USA., 2005.
[16] C. Jones, “Assessment and Control of Software Risks”, Yourdon Press, 1994.
[17] G. Sifri, “Accurate Estimates Critical for Software Development Projects”, ESI International, Inc., 2001
[18] J. A. Ward, “Productivity Through Project Management: Controlling the Project Variables”, Information Management, 1994.
[19] Construx, “10 Deadly Sins of Software Estimation”, Construx Software Builders, Inc., 2002.
[20] C. Simons, “Software Sizing and Estimating: MK II”, John Wiley & Sons, 1991.
[21] T. Menguy, “Concrete Estimation: Size, Effort, Schedule”, http://www.essi.fr/~hugues/GL/Projec t/ estimation.html
[22] C. F. Kemerer, “An Empirical Validation of Software Cost Estimation Models”, Communication ACM, vol.30, no.5, pp. 416-429, 1987.
[23] C. Jones, “Applied Software Measurement: Assuring Productivity and Quality”, McGraw-Hill, 1991.
[24] S. McConnell, “Rapid Development”, Microsoft Press, 1996.
[25] ISBSG, “Worldwide Software Development - The Benchm ark Release 8”, Victoria, Australia International Software Benchmarking Standards Group, 2004.

저 자 소 개

이 상 운 (Sang-Un, Lee)

1983년~1987년 : 한국항공대학교 항공전자공학과 (학사)
1995년 ~ 1997년 : 경상대학교 컴퓨터과학과 (석사)
1998년 ~ 2001년 : 경상대학교 컴퓨터과학과 (박사)
2003년 : 강원도립대학 컴퓨터응용과 전임강사
2004년 ~ 2007.2 : 국립 원주대학 여성교양과 조교수
2007.3 ~ 현재 : 강릉원주대학교 과학기술대학 멀티미디어공학과 부교수
관심분야 : 소프트웨어 프로젝트 관리, 소프트웨어 개발 방법론, 소프트웨어 척도, 분석과 설계 방법론, 소프트웨어 시험 및 품질보증, 소프트웨어 신뢰성, 신경망, 뉴로-퍼지, 그래프 알고리즘
e-mail : sulee@gwnu.ac.kr

최 명 복 (Myeong-Bok, Choi)

1992년 : 호서대학교 전자계산학과(학사)
1994년 : 아주대학교 컴퓨터공학과(석사)
2001년 : 아주대학교 컴퓨터공학과(박사)
1997~현재 : 강릉원주대학교 멀티미디어공학과(교수)
2004. 1~현재 : 한국인터넷방송통신학회 이사
관심분야 : 지능형 정보검색, 퍼지응용, 지식표현, 신경망, 지능형 교통제어, 소프트웨어 공학, 알고리즘
e-mail : cmb5859@gmail.com, cmb1@gwnu.ac.kr