Gompertz 성장곡선을 이용한 소프트웨어 프로젝트의 개발 성공률과 완료율 추정

이 상 운†

요 약

소프트웨어 복잡도가 증가할수록 소프트웨어 성공률은 기하급수적으로 감소하며, 반대로 실패율은 증가한다. 소프트웨어 규모 증가에 따른 실패율은 성장곡선으로 표현할 수 있다. 이 현상에 따라, 본 논문은 Gompertz 성장곡선으로 개발 성공률과 완료율을 추정하였다. 먼저, 수치적으로 제시된 10^n의 소프트웨어 규모를 로그 값으로 변환시켜 데이터 간격을 일정하게 하였다. 로그값의 소프트웨어 규모 변화에 따른 개발 성공률과 완료율의 함수관계를 유도하고자 하였다. 그러나 이 관계를 적절히 표현하는 함수를 찾지 못하였다. 따라서 본 논문에서는 개발 성공률의 역 개념인 실패율과 완료율의 역 개념인 취소율을 도입하였다. 로그값의 소프트웨어 규모 변화에 따른 개발 실패율과 취소율 관계는 성장곡선 형태를 나타내었다. 결론적으로, 개발 취소율과 실패율을 적절히 표현하는 함수로 Gompertz 성장곡선을 적용한 결과 실측 데이터를 적절히 표현할 수 있었다. 본 모델을 적용하면 특정 규모의 소프트웨어에 대한 개발 성공률과 완료율을 보다 정확히 얻을 수 있을 것이다.

키워드 : 프로젝트 성공률, 프로젝트 완료율, 복잡도, 소프트웨어 규모, 성장곡선

Estimation of Software Project Success and Completion Rate Using Gompertz Growth Function

Sang-Un Lee†

ABSTRACT

As the software complexity increases, the development success rate decreases and failure rate increases exponentially. The failure rate related to the software size can be described by a growth function. Based on this phenomenon, this paper estimates the development success and completion rate using the Gompertz growth function. At first, we transformed a software size of numerically suggested 10^n into a logarithm and kept the data interval constantly. We tried to derive a functional relationship between the development success rate and the completion rate according to the change of logarithmic software size. However, we could not find a function which can represent this relationship. Therefore, we introduced the failure rate and the cancel rate which are inverse to the development success rate and completion rate, respectively. Then, we indicated the relation between development failure rate and cancel rate based on the change of software size, as a type of growth function. Finally, as we made the Gompertz growth function with the function which describes the cancel rate and the failure rate properly. We could express the actual data suitably. When you apply the growth function model that I suggested, you will be able to get the success rate and completion rate of particular size of software very accurately.

Key Words : Project Success Rate, Complete Rate, Complexity, Software Size, Growth Curve

1. 서 론

일반적으로 사용자는 많은 양의 기능을 요구하고, 고객은 소프트웨어 개발에 소요되는 비용과 시간을 최소화시키기를 원한다. 이에 따라 개발될 소프트웨어의 복잡성 증가, 고객의 높은 기대와 계속되는 요구사항의 변경으로 인해 고질적으로 소프트웨어 위기(Software Crisis)가 해소되지 않고 있다. 소프트웨어 위기는 일반적으로 대부분의 프로젝트들이 예산 초과, 일정 지연, 요구사항 불만족과 낮은 품질의 제품을 고객에게 인도함에 따른 프로젝트 개발 실패를 의미한다.

소프트웨어의 위기를 극복하고자 많은 개발 방법론과 프로세스들이 발전되어 왔다. 그러나 지금까지도 소프트웨어의 비가시성, 복잡성과 유연성 특성으로 인해 소프트웨어 관리자들은 항상 위기에 직면해 있다. 이에 따라 프로젝트를 신규로 수주하였을 경우, 프로젝트 관리자는 개발 성공 확률과 프로

† 정 회 원 : 국립 원주대학 여성교양과 조교수
논문접수 : 2006년 3월 31일, 심사완료 : 2006년 7월 4일

젝트 완료 확률 정보를 사전에 인지하고 프로젝트를 관리하는 능력이 필수적으로 요구된다.

Standish Group의 CHOAS 보고서[1-4]는 소프트웨어 성공률과 프로젝트 완료율에 대해 개발 업체의 규모에 기반하여 개략적인 수치만을 제공하고 있다. 그러나 하나의 개발 조직의 경우에도 수행되는 프로젝트 규모가 소형, 중형, 대형에 상관없이 성공할 확률이 동일하지 않다. 또한, Schwaber[5]는 프로젝트 복잡도가 증가할수록 개발 성공률은 기하급수적으로 감소한다는 개략적인 프로파일만을 제시하고 있다. 따라서 이 보고서를 일반적으로 적용하는데 문제가 발생한다.

지금까지의 연구 결과로 볼 때, 프로젝트 규모가 변화함에 따라 개발 성공률이 어떠한 분포를 나타내는지 알 수 없다. 이로 인해 신규로 개발될 소프트웨어 규모가 추정되었을 때, 이 프로젝트의 개발 성공률을 추정할 수 있는 정보를 얻지 못하여 프로젝트 계획 단계에 반영시킬 수 없다.

본 논문은 소프트웨어 규모에 기반하여 개발 성공률과 완료율을 추정할 수 있는 통계적 모델을 유도하고 제안된 모델을 평가해 본다. 먼저, 소프트웨어 개발 성공률에 대한 기준 설정, 소프트웨어의 크기를 결정하는 규모에 대해 살펴보고, 소프트웨어 개발 성공률 분포를 유도한다. 다음으로 소프트웨어 성공률과 완료율을 추정할 수 있는 분포를 유도하고, 이 분포에 적합한 통계적 함수를 선택하여 제안된 모델의 적합성을 평가한다.

2장에서는 소프트웨어 성공률과 성공률 프로파일에 관련된 연구 결과를, 3장에서는 성공률을 추정하는 분포를 유도하고, 이에 적합한 통계적 모델을 선정한다. 4장에서는 실측 데이터에 적용하여 제안된 모델의 적합성을 평가한다.

2. 관련 연구

대표적인 소프트웨어 개발 성공에 대한 분석은 Standish Group의 CHAOS 보고서[1-4]가 있다. 이 보고서는 주어진 개발 일정(On-Time) 내에서, 주어진 예산(On-Budget)으로 개발 초기에 확정된 고객 요구사항인 기능성(Functionalities)과 형상(Features)을 만족하느냐로 개발의 성공 기준을 설정하고 있다. 이 보고서는 프로젝트를 3가지 경우로 분류하였다.

(1) 프로젝트 성공(Project Success) : 초기에 명시된 모든 기능을 만족하는 소프트웨어를 주어진 예산 범위 내에서 주어진 일정에 맞추어 완료한 경우
(2) 프로젝트 의심(Project Challenged) : 프로젝트가 완료되어 운영되고 있지만 예산과 일정을 초과하였고 초기에 명시된 기능을 만족시키지 못하는 경우
(3) 프로젝트 취소(Project Impaired 또는 Canceled) : 프로젝트가 완료되기 이전의 개발 과정에서 취소된 경우

Standish Group Report[1-4]가 제시하고 있는 년도별 프로젝트 수행 결과는 <표 1>에 제시되어 있다. 표에서 알 수 있듯이 IT 프로젝트 성공률은 약 30% 정도로 일반적으로 알려져 있다.[6]

〈표 1〉 프로젝트 성공, 의심과 취소 경향

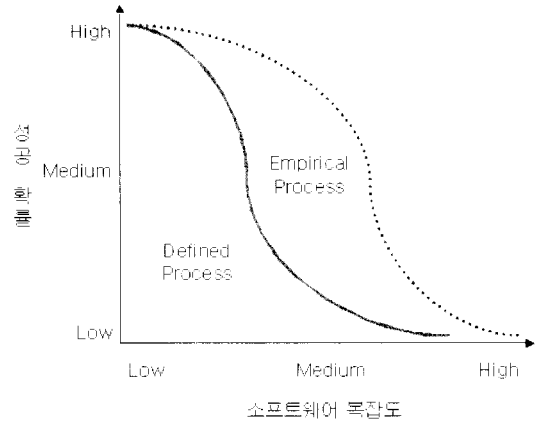
년도	성공	의심	취소
1994	16%	53%	31%
1996	27%	33%	40%
1998	26%	46%	28%
2000	28%	49%	23%
2003	34%	51%	15%
2004 3/4분기	29%	53%	18%

Standish Group 보고서[1]는 프로젝트 수행 결과를 업체 규모(대형, 중형, 소형)에 기반하여 분류하였다. 그러나 임의의 업체라도 수행되는 프로젝트의 규모에는 많은 차이가 있을 수 있어 이 보고서를 일반적으로 적용하는데 문제가 발생한다. 왜냐하면, "어느 한 부류에 속한 업체(예로 대형 업체)에서 수행되는 프로젝트 규모가 소형, 중형, 대형에 상관없이 성공할 확률은 9% 이다" 라는 결론을 내릴 수는 없기 때문이다. 따라서 개발업체 규모보다 일반적인 측도(Measure)에 기반하여 프로젝트의 성공, 의심과 취소 비율을 얻는 것이 프로젝트 관리를 위해 보다 유용한 정보가 될 수 있다.

소프트웨어를 개발하는데 소요되는 노력(Effort), 비용(Cost)과 기간(Duration)을 예측하기 위해서는 소프트웨어를 정량화할 수 있는 근본적인 단위(Fundamental Unit)가 필요하다. 소프트웨어 측정분야에서는 규모(Size)를 소프트웨어의 근본적인 단위로 채택하였다[8]. 소프트웨어 규모를 표현할 수 있는 속성으로 Mišić[8]과 Mcphee[9]는 복잡도(Complexity), 기능성(Functionality), 길이(Length)와 재사용(Reuse)으로, Sultanoglu[10], Penton et al.[11]는 복잡도, 기능성과 길이로 정의하였다. 길이에 기반한 소프트웨어 규모 추정 방법은 라인 수(Lines Of Code, LOC)로 측정한 것이며, 대표적으로 COCOMO 모델[12]이 있다. 기능에 기반한 규모 추정방법으로는 기능점수(Function Point, FP), 완전 기능점수(Full Function Point, FFP), COCOMO II, 유스케이스 점수(Use Case Point, UCP), 특징점수(Feature Point, FePs)와 객체점수(Object Point, OP) 등이 있다[13-17]. 소프트웨어 규모 추정 기법들이 개발 방법론의 채택과 더불어 기능 요소와 복잡도를 모두 고려하는 방법으로 변화되고 있다.

소프트웨어를 납품하는 개발팀의 능력(Capability)은 적시에 경제적 효과를 가지고 소프트웨어를 납품하기 위한 개발 조직의 능력에 영향을 미치는 다양한 위험인자(Risk Factor)들인 소프트웨어 프로세스, 개발조직의 숙련도 수준, 자동화, 개발 조건과 비즈니스 환경의 영향 등이 있다.

개발하고자 하는 소프트웨어의 규모가 추정되었을 경우, "주어진 개발일정과 예산 범위 내에서 소프트웨어의 개발에 성공할 확률은 얼마인가?", "주어진 예산과 개발 일정을 초과하더라도 개발을 완료할 수 있는 확률은 얼마인가?" 등은 프로젝트를 보다 효율적으로 관리하는데 필요한 정보이다. 이에



(그림 1) 복잡도-성공률 관계

대한 전형적인 프로파일을 살펴보자.

(그림 1)은 소프트웨어의 복잡도가 증가함에 따라 전형적인 개발 성공률의 경향을 나타내고 있다[5]. 이 그림은 폭포수, 나선형 또는 반복 프로세스를 사용하는 현재의 개발 환경인 정의된 프로세스(Defined Process)와 기민한 방법론(Agile Methodology) 중 SCRUM[18]인 경험적 프로세스(Empirical Process)에 대한 사례이다.

소프트웨어 복잡도가 증가할수록 예측할 수 없는 환경에 적응할 수 있는 유연성이 저하됨에 따라 성공률이 급격히 감소하는 경향을 보이고 있다. 소프트웨어의 규모가 커질수록 복잡도 역시 기하급수적으로 증가한다. 왜냐하면 소프트웨어 규모 측정 기법은 복잡도 까지 고려하고 있기 때문이다. 따라서 (그림 1)의 가로축을 소프트웨어 규모로 생각할 수 있으며, 소프트웨어 규모에 따른 개발 성공 확률로 고려할 수 있다.

<표 2>는 소프트웨어 규모가 증가함에 따른 프로젝트의 성공과 실패의 전반적인 패턴을 나타내고 있는 실측 데이터이다[19-22]. 여기서, Jones[19] 데이터는 1996년도에, Jones [22]는 2000년도에 발표된 자료로, 수집된 데이터들의 차이로 인해 개발 확률에 변동이 발생하고 있다. 따라서 최근의 데이터인 Jones[22]의 데이터를 활용하는 것이 보다 타당할 것으로 판단된다. 그러나 Jones[19]의 데이터인 경우, 2003년과 2004년에 발표된 Capers[20]과 Orr[21] 논문에서 인용된 사례를 볼 때 아직까지는 유용하게 사용될 수 있다고 판단되어 본 논문에서 실험 데이터로 적용하고자 한다.

〈표 2〉 프로젝트 성공과 실패 패턴

(a) Jones[19], Capers[20]와 Orr[21] 제시 데이터

소프트웨어 규모 (기능점수, FP)	프로젝트 개발 확률(%)			
	Early Time	On Time	Delayed Time	Canceled
1	14.68	83.16	1.92	0.25
10	11.08	81.25	5.67	2.00
100	6.06	74.77	11.83	7.33
1000	1.24	60.76	17.67	20.33
10000	0.14	28.03	23.83	48.00
100000	0.00	13.67	21.33	65.00

(b) Jones[22] 제시 데이터

소프트웨어 규모 (기능점수, FP)	프로젝트 개발 확률(%)			
	Early Time	On Time	Delayed Time	Canceled
1	6.00	92.00	1.00	1.00
10	8.00	89.00	2.00	1.00
100	7.00	80.00	8.00	5.00
1000	6.00	60.00	17.00	17.00
10000	3.00	23.00	35.00	39.00
100000	1.00	15.00	36.00	48.00

3. 소프트웨어 성공률 추정 모델

3.1 소프트웨어 성공률 분포 유도

<표 2>에서 계약기간 이전(Early Time)에 개발을 성공할 확률을 $P(e)$로, 계약기간에 맞추어(On Time) 개발을 완료한 확률을 $P(o)$, 계약기간을 초과(Delayed Time) 하였지만 개발을 완료한 확률을 $P(d)$, 개발이 취소되는 확률을 $P(c)$라 하자. 프로젝트 관리자의 관심은 개발될 프로젝트가 성공할 확률과 일정을 초과하더라도 개발을 완료할 수 있는 확률이 될 수 있다.

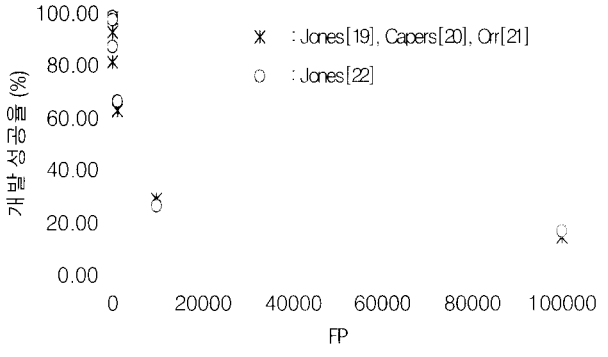
개발 완료율(Completion Rate)을 $P(co)$, 개발 취소율(Canceled Rate)을 $P(ca)$, 개발 성공률(Success Rate)을 $P(su)$, 개발 실패율(Failure Rate)을 $P(fa)$라 하자. 이 가정에 따르면 $P(co) = P(e) + P(o) + P(d)$, $P(ca) = P(c)$, $P(su) = P(e) + P(o)$, $P(fa) = P(d) + P(c)$가 된다. <표 2>의 데이터를 이 가정에 기반하여 다시 계산하면 <표 3>과 같다.

<표 3>의 개발 성공률과 개발 완료율을 표현할 수 있는 적절한 모델을 찾는다면 신규로 개발될 소프트웨어 규모에 대한 개발 성공률과 완료율을 알 수 있을 것이다. 모델을 찾는 과정은 다음과 같이 수행되었다.

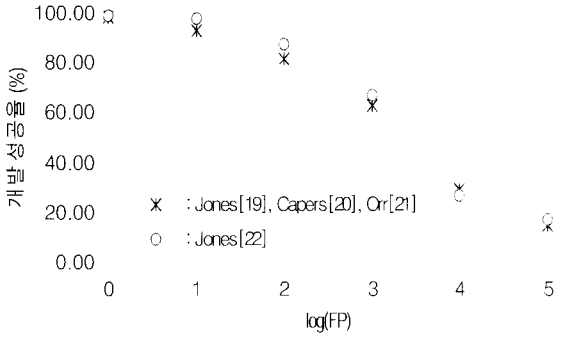
Step 1. *독립변수인 소프트웨어 규모(기능점수)를 가로축으로 하고, 종속변수인 개발 성공률 $P(su)$를 세로축으로 하여 값을 표시하고 이 값 들을 연결한 그래프를 그려 적절한 함수를 찾는다.*

〈표 3〉 개발 성공과 개발 완료 패턴

기능점수 (FP)	개발 성공률 $P(su)$		개발 완료율 $P(co)$		개발 취소율 $P(ca)$	
	Jones[19], Capers[20], Orr[21] 데이터	Jones [22] 데이터	Jones[19], Capers[20], Orr[21] 데이터	Jones [22] 데이터	Jones[19], Capers[20], Orr[21] 데이터	Jones [22] 데이터
1	97.84%	98.00%	99.76%	99.00%	0.25%	1.00%
10	92.33%	97.00%	98.00%	99.00%	2.00%	1.00%
100	80.83%	87.00%	92.66%	95.00%	7.33%	5.00%
1000	62.00%	66.00%	79.67%	87.00%	20.33%	17.00%
10000	28.17%	26.00%	52.00%	61.00%	48.00%	39.00%
100000	13.67%	16.00%	35.00%	52.00%	65.00%	48.00%



(그림 2) 소프트웨어 규모와 개발 성공률 관계



(그림 3) log(FP)와 개발 성공률 관계

<표 3>에 대한 Step 1의 결과는 (그림 2)에 제시되어 있다. 그림으로부터 규모 변화에 따른 개발 성공률은 지수분포를 따르는 것으로 생각할 수 있다.

그러나 기능점수가 10^n으로 증가함으로 데이터 간격이 일정하지가 않고 규모가 작은 부분에 편중된 결과를 나타내고 있다. 따라서 소프트웨어 규모 간격을 일정하게 분포시키는 로그함수를 고려할 수 있다.

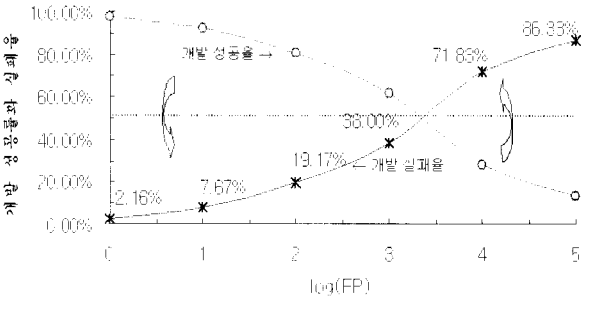
Step 2. *가로축의 데이터 간격을 일정하게 하기 위해 소프트웨어 규모(FP)에 log를 취한* $\log(FP)$*로 치환한다.*

<표 3>의 log(FP) 변화에 따른 개발 성공률은 (그림 3)에 제시하였다. 이는 Schwaber[5]가 제시한 소프트웨어 복잡도와 성공률 관계인 (그림 1)의 형태와 일치하는 결과이다.

(그림 3)을 적절히 표현할 수 있는 통계적 분포 함수를 찾기가 쉽지 않다. 따라서 기존에 제시된 통계적 함수로 변환하는 과정이 필요하다.

Step 3. *세로축의 개발 성공률을 역 개념인 개발실패율로 치환하여 그래프를 그린다.*

(그림 3)에서 "1-개발 성공률"은 개발 실패율 $P(fa)$가 된다. 즉, $P(fa)=1-P(su)$이다. 이러한 관계로부터 <표 3>의 데이터를 표현한 결과는 (그림 4)에 제시되어 있다.



(그림 4) 개발성공률과 개발 실패율

개발 실패율 $P(fa)$ 그래프는 S자형의 성장곡선(Growth Curve) 형태를 따르므로 성장곡선 모델을 적용하여 모델을 얻은 후 $1-P(fu)$로 소프트웨어의 규모에 따른 개발 성공률 $P(su)$을 얻을 수 있다. 개발 완료율도 역 개념인 개발 취소율을 도입하면 동일한 방법으로 성장곡선 형태로 표현된다.

3.2 성장곡선을 이용한 성공률과 완료율 추정

현존하는 실세계의 많은 성장 현상은 초기에는 점진적으로 증가하다가 중간 구간에서 보다 빠르게 증가하고 한계점으로 도달하면서 성장 속도가 느려지며 일정한 시간 구간 후에는 최대값에서 수평이 되는 S자 패턴을 보이고 있다. 따라서 성장 곡선을 일명 S자(S shaped) 곡선, 시그모이드 (Sigmoidal) 곡선 또는 학습(Learning) 곡선이라 부르며 생명체의 성장이나 인간의 기술 성취도의 닮은 점을 표현하고 있다[23].

성장곡선을 표현하기 위해 다양한 수학적 함수가 제안되었다. 대표적인 성장곡선으로는 대칭 로지스틱 성장 곡선, 일반화된 로지스틱 곡선, Pearl 곡선과 Gompertz 곡선 등이 있다[23]. 대칭 로지스틱 성장곡선과 일반화된 로지스틱 곡선은 생물체의 성장패턴을 보형화하는데 널리 적용되고 있다. Pearl 곡선은 미국 통계학자인 Raymond Pearl24]이 제안한 것으로 인구예측에 적용되며, 그래프가 중간지점을 경계로 상호 대칭이 되는 특징을 갖고 있다. 이에 비해, Gompertz 곡선은 영국의 공증인이자 수학자인 Benjamin Gompertz[25]에 의해 제안되었으며, 기술예측분야에 적용되고 있다. 소프트웨어 규모인 $\log(FP)$와 프로젝트 실패율 $P(fa)$의 관계는 식 (1) 또는 식 (2)의 곡선으로 표현된다. 이 곡선은 Pearl 곡선과 닮았지만, 곡선이 비대칭이 되는 특징을 갖고 있다.

$$P(fa)=a\cdot b^{c^{\log(FP)}} \quad (1)$$

$$P(fa)=a\cdot e^{-b\cdot c^{-\log(FP)}} \quad (2)$$

소프트웨어 개발 성공률을 추정함에 있어 (그림 4)는 규모 $\log(FP)$의 중간 지점에서 개발 실패율 $P(fa)$가 좌·우 대칭이 되지 않기 때문에 Gompertz 성장곡선을 적용할 수 있다.

Gompertz 성장곡선을 이용하여 성공률과 완료율을 추정하

기 위해서는 실측 데이터를 활용하여 모수 a, b, c의 값을 구해야만 한다. 본 논문에서는 식 (2)를 적용하고자 한다. 식 (2)를 적용한 이유는 다음과 같다. 식 (1)에 log를 취하면$log(a) + c^{log(FP)} \cdot log(b)$가 된다. 여기서 $log(b) = b$로 치환하면 $log(a) + b \cdot c^{log(FP)}$가 된다. 식 (2)에 log를 취하면 $log(a) - bc^{\cdot log(FP)}$가 되며, 여기서 $e^c = c$로 치환하면 $log(a) + b \cdot c^{log(FP)}$가 되어 식 (1)과 (2)는 동일한 식이 된다. 또한, 식 (1)의 모수 a, b, c를 추정하기 위해서는 최소 자승법(LMS, Least Mean Squares)이나 최우 추정법(MLE, Maximum Likelihood Estimation) 등 다양한 방법을 이용할 수 있다. 그러나 이 모델들은 비선형으로 초기 값을 적절히 설정하는데 어려움이 있다[26]. 반면에 식 (2)는 두 번의 자연로그 변수변환을 거치면 식 (3)의 선형 방정식이 얻어지며 이로부터 상용 통계 패키지 도구를 이용하여 선형회귀분석을 수행하면 모수들의 값을 쉽게 구할 수 있기 때문이다.

$$\ln[-\ln(\frac{P(fa)}{a})] = \ln(b) - c \cdot \log(FP) \quad (3)$$

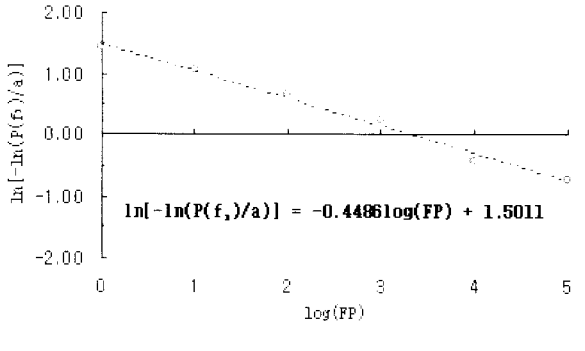
<표 3>의 데이터에 대해 $\ln[-\ln(\frac{P(fa)}{a})]$의 그래프가 선형이 되도록 a 값을 변화시키면 모수 b, c의 값을 쉽게 얻을 수 있다. 따라서 LMS와 MLE 방법을 적용하지 않고 식 (3)의 변수변환을 통한 선형식에 대해 실제 데이터를 적용하여 회귀분석을 수행하였다.

4. 실측 데이터 적용 및 평가

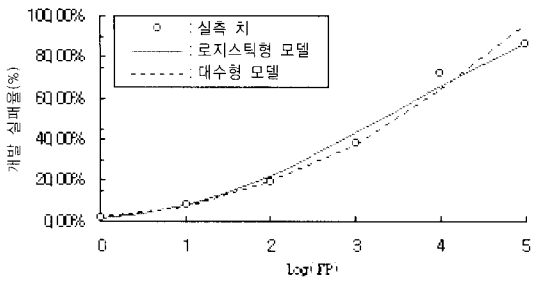
<표 3> 데이터를 적용하여 개발 성공률과 개발 완료율을 구하는 모델을 평가해 본다.

4.1 개발 성공률 추정 모델

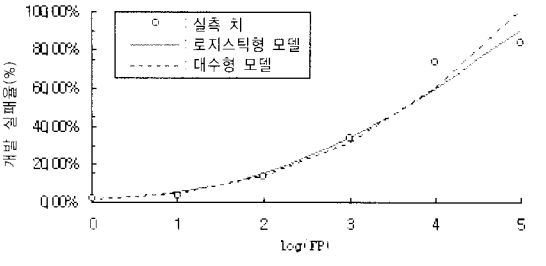
<표 2>의 Jones[19], Capers[20]와 Orr[21] 데이터에 대해 $log(FP)$의 6개 값에 대한 식 (3)의 $\ln[-\ln(\frac{P(fa)}{a})]$가 선형이 되도록 a 값을 조절한 결과 $a = 139$에서 (그림 5)의 선형 회귀식을 얻었다.



(그림 5) 개발 실패율 모수 추정 선형회귀분석



(a) Jones[19], Capers[20]와 Orr[21] 데이터



(b) Jones[22] 데이터

(그림 6) 개발 실패율 추정 결과

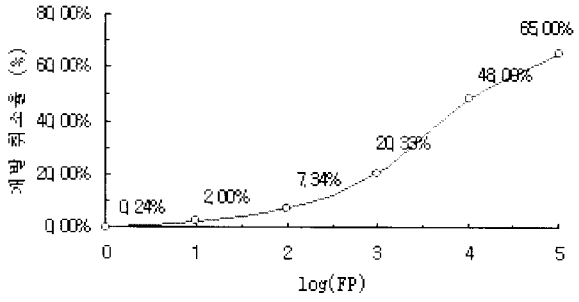
식 (3)에서 $c = 0.4486$를 알 수 있으며, $\ln(b) = 1.5011$로부터 $b = e^{\ln b} = 4.4866$을 얻을 수 있다. 따라서 개발 실패율 추정 값은 식 (2)에 의해 $P(fa) = 139e^{-4.4866e^{-0.4486\log(FP)}}$ 모델로 계산된다.

Jones[22] 데이터는 $P(fa) = 255e^{-5.4265e^{-0.5311\log(FP)}}$를 얻었다. 이와 같이 추정된 모델의 값을 이용해 실측값과 비교하여 (그림 6)에 나타내었다.

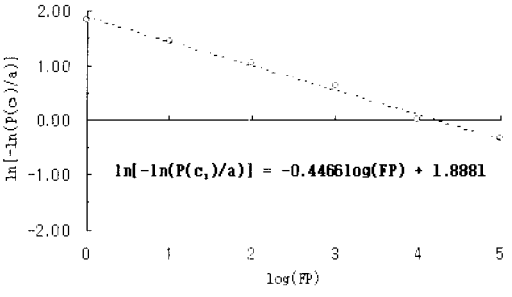
4.2 개발 완료율 추정 모델

<표 3> 데이터의 개발 완료 확률 $P(co)$로 부터 "1-$P(co)$"을 개발 취소율 $P(w)$라 하자. 독립변수인 소프트웨어 규모 $\log(FP)$를 가로축에, 종속변수인 개발 취소율 $P(ca)$를 세로축으로 하여 그린 결과는 (그림 7)에 제시되어 있다.

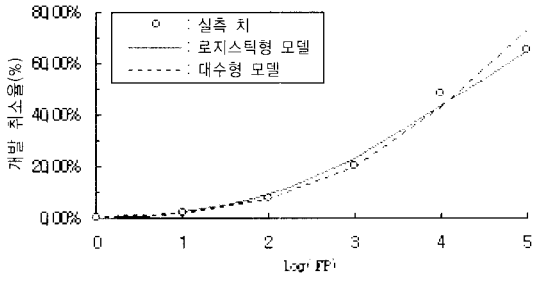
식 (6)과 (7)에서 개발 실패율 $P(fa)$를 개발 취소율 $P(ca)$로 치환하면 개발 완료율을 추정할 수 있다. Jones[19], Capers[20]와 Orr[21]의 (그림 7) 데이터에 대해 자연로그를 2번 취한 식 (3)에 따라 a 값을 조절한 후 선형 회귀분석을 수행한 과정은 (그림 8)에 제시되어 있으며, $a = 132$이다.



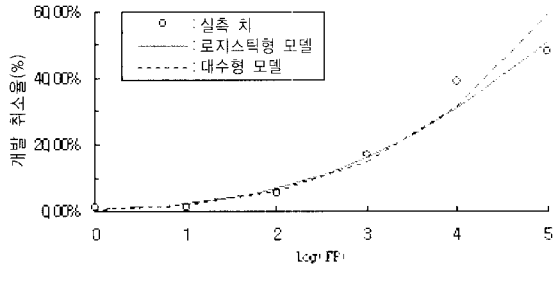
(그림 7) log(FP)에 따른 개발 취소율



(그림 8) 개발 취소율 모수 추정 선형회귀분석



(a) Jones[19], Capers[20]와 Orr[21] 데이터



(b) Jones[22] 데이터

(그림 9) 개발 취소율 추정 결과

$\ln(b)=1.8881$로 부터 $b=6.6068$을 얻어 개발 취소율에 대한 추정 값은 $P(ca)=132e^{-6.6068e^{-0.4466\log(FP)}}$ 모델로 계산된다. Jones[22] 데이터는 $P(ca)=225\cdot e^{-6.2189e^{-0.5510\log(FP)}}$를 얻었다. 추정된 모델의 값을 이용해 실측값과 비교하여 (그림 9)에 나타내었다.

4.3 모델 검증 및 성능 평가

제안된 모델들의 회귀분석 결과가 유효한지 검증하기 위해, 개발 실패율과 취소율 추정 모델의 일원 분산분석(One-way Analysis Of Variance) 결과를 <표 4>에 제시하였다.

〈표 4〉 분산분석표

(a) 개발 실패율 추정 모델

모델	자유도	제곱합	평균제곱합	F	$F(\alpha)$
회귀	1	0.5972	0.5972	359.9822	4.55E-05
잔차	4	0.0066	0.0017		
계	5	0.6038			

(b) 개발 취소율 추정 모델

모델	자유도	제곱합	평균제곱합	F	$F(\alpha)$
회귀	1	0.3569	0.3569	534.8905	2.07E-05
잔차	4	0.0027	0.0007		
계	5	0.3596			

식 (3)에서 종속변수인 $\ln[-\ln(\frac{P(fa)}{a})]$는 독립변수인 $log(FP)$ 1개에만 영향을 받으므로, "회귀"에 적용된 자유도(df, degree of freedom)는 1이 된다. 또한, <표 2>에서 전체 데이터(N)는 6개이므로, "계"의 자유도는 회귀에 적용된 1개를 N-1=5가 되며, "잔차"에 적용된 자유도는 5-1=4가 된다. 또한, F는 분산비이며, $F(\alpha)$는 기각역이다. 즉, 표에서 개발 실패율과 취소율 추정 모델들은 모두 유의수준 α에서 $F>F_{\alpha}$가 되어 분산비가 기각역 범위에 해당되지 않으므로 회귀 분석이 유의함을 알 수 있다.

모델의 성능을 평가하는 기준으로 회귀분석 결과 얻어진 결정계수(Coefficient of determination, $0\le R^2\le 1$)와 더불어 모델의 정확도를 판단하기 위해 MMRE (Mean Magnitude Relative Error)와 Pred(0.25)를 분석하였다. 종속변수의 값은 독립변수에 의해 결정되는 부분과 미지의 오차의 합으로 나타나며, 총 변동을 설명하는데 있어서 회귀직선에 의해 설명되는 변동 비율을 결정계수라 하며, 값이 클수록 회귀식의 설명력이 크다. 그러나 좋은 모델을 선정할 수 있는 결정계수에 대한 기준은 없는 실정이다. MMRE는 다음과 같이 측정된다. 상대오차(RE, Relative Error) = (추정치 - 실측치)/실측치, MRE(Magnitude of RE) = $|RE|$, MMRE(Mean MRE) = $100/n*\sum_{i=1}^{n} MRE$. MMRE가 작은 값이면 평균적으로 좋은 모델임을 알 수 있다. Pred(0.25)는 예측된 값의 MRE가 실측값의 MRE ±25% 범위에 있는 개수를 의미한다. Conte et al.[27]는 $MMRE\le 0.25$ (25% 이하)이면 개발비용을 예측하는 모델로 채택 가능하며, Pred(0.25)가 75% 이상일 때 좋은 예측모델로 될 수 있음을 제안하였다.

소프트웨어 규모에 따라 개발 실패율과 취소율을 구할 수 있는 모델이 없는 관계로 본 제안 모델과 타 모델과의 성능 비교는 불가능하다. 따라서 제안 모델의 결정계수, MMRE와 Pred(0.25)를 분석한다.

결정계수, MMRE와 Pred(0.25)를 분석한 결과는 <표 5>에 제시되어 있다.

〈표 5〉 모델 성능

구분	개발실패율 추정		개발 취소율 추정	
	Jones[19], Capers [20], Orr[21] 데이터	Jones[22] 데이터	Jones[19], Capers [20], Orr[21] 데이터	Jones[22] 데이터
결정계수	0.9863	0.9539	0.9907	0.9562
MMRE	11.54%	27.21%	12.23%	38.72%
Pred(0.25)	0.8333	0.6667	0.8333	0.5000

개발 실패율과 개발 취소율을 추정하는 모델들에 있어서 공통적으로 MMRE와 Pred(0.25)는 좋은 결과를 나타내었다. MMRE의 경우, Conte et al.[27]의 기준을 모두 만족하는 결과를 나타내었다. 그러나 Pred(0.25) 기준에 대해서는 Jones[19], Capers[20]와 Orr[21] 데이터는 만족하나 Jones[22] 데이터는 불만족하는 것으로 판명되었다. 결론적으로 Gompertz 성장곡선 모델을 적용하여 프로젝트의 성공률과 완료율을 추정할 수 있을 것이다.

5. 결론 및 향후 과제

소프트웨어의 위기를 극복하고자 많은 개발 방법론과 개발 프로세스들이 발전되어 왔으나 아직까지도 소프트웨어 관리자들은 항상 이러한 위기에 직면해 있다. 소프트웨어 프로젝트를 신규로 수주하였을 경우 개발을 성공할 확률과 프로젝트를 완료할 수 있는 확률이 얼마인지에 대한 정보는 프로젝트 관리자에게 필수적인 요소이다. 그러나 프로젝트 규모에 기반하여 개발 성공률을 통계적 모델을 이용하여 제시한 이론은 없는 실정이다.

본 논문은 소프트웨어 규모에 기반하여 개발 성공률과 완료율에 대한 분포를 유도하고, 이에 적합한 통계적 함수를 선정하여 모델의 적합성을 검증하였다. 제안된 이론을 기반으로 하면 개발될 소프트웨어 프로젝트의 규모만 알고 있으면 개발 성공률과 완료율에 대한 정보를 쉽게 구할 수 있다.

그러나 제안된 모델 제시에 적용된 표본의 개수가 적어 일반화된 모델 제시는 되지 않은 부족함이 있다. 또한, 제안된 모델은 소프트웨어의 개발 일정에만 기반하고 있어 개발 일정과 개발비용 모두를 고려한 모델 연구가 필요하다. 따라서 추후 이 분야에 대한 심층적인 연구로 보다 일반화된 개발 성공률 추정 모델이 제시될 것이다.

참 고 문 헌

[1] Standish group, "The CHAOS Report,"http://www.standishgroup.com/sample/PDFpages/chaos1994.pdf, 1994.

[2] Standish group, "CHAOS: A Recipe for Success," http://www.standishgroup.com/sample/PDFpages/chaos1999.pdf, 1999.

[3] Standish group, "Extreme CHAOS," http://www. standishgroup.com/sample/PDFpages/extreme_chaos.pdf, 2001.

[4] Standish group, "CHAOS Demographics and Project Resolution,"http://www.standishgroup.com/sample/PDFpages/ q3_spotlight.pdf, 2004.

[5] K. Schwaber, "Scaling Agile: It Depends on Common Sense," CAMUG Meeting, 2003.

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

[7] D. Garmus and D. Herron, "Estimating Software Earlier And More Accurately," David Consulting Group, http://www.davidconsultinggroup.com/articles/pbestart.html

[8] V. B. Mišić, "Software Size and Cost Estimation," Department of Computer Science, University of Belgrade, 2003.

[9] C. Mcphee, "SENG 621-Software Process Management," University of Calgary, 1999.

[10] S. Sultanoglu, "Software Measurement," Department of Computer Science & Eng., Hacettepe University, 1998.

[11] N. E. Fenton and S. L. Pfleegr, "Software Metrics: A Rigoous and Practical Approach, 2nd Edition, PWS Publishing Company, 1997.

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

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

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

[15] ISO/IEC FDIS 19761, "Software Engineering- COSMIC-FFP-A Functional Size Measurement Method," 2002.

[16] B. W. Boehm et al, "Software Cost Estimation with COCOMO II," Prantice-Hall, 2000.

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

[18] J. Bach, "SCRUM Software Development Process: Building The Best Possible Software," Advanced development Methods, 1995.

[19] C. Jones, "Patterns of Software Systems Failure and Success," International Thomson Press, 1996.

[20] J. Capers, "Why Flawed Software Projects Are Not Cancelled in Time," Cutter IT Journal, Vol.16, No.12, pp.12-17, 2003.

[21] K. Orr, "Pushing the Envelope: Managing Very Large Projects," Cutter Agile Software Development & Project Management Advisory Service Executive Report, Vol.5, No.7, 2004.

[22] C. Jones, "Software Assessment, Benchmark, and Best Practices," Information Technology Series, Addison-Wesley, 2000.

[23] C. Henry, "The Growth Curve," http://www.anzpug.org/sp/index.jsp, PRIMAVERA Users Groups, Technology and Operations Management, California Polytechnic and State University.

[24] R. Pearl, "The Biology of Population Growth," New York: Knopf, 1978.

[25] B. Gompertz, "On The Nature of The Function Expressive of The Law of Human Mortality, and on a New Mode of Determining the Value of Life Contingencies," Phil. Trans. Roy. Soc. London. Vol.123, pp.513-585, 1832.

[26] Weibull.com, "Software Reliability Growth Model," http://www.weibull.com/relgrowthwebcontents.html

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

이 상 운

e-mail : sulee@sky.wonju.ac.kr

1983년~1987년 한국항공대학교 항공전자공학과(학사)

1995년~1997년 경상대학교 컴퓨터과학과(석사)

1998년~2001년 경상대학교 컴퓨터과학과(박사)

1992년~2002년 국방품질관리소 항공전자장비 및 소프트웨어 품질보증 담당

2003년 강원도립대학 컴퓨터응용과 전임강사

2004년~현 재 국립 원주대학 여성교양과 조교수

관심분야 : 소프트웨어 프로젝트 관리, 소프트웨어 개발 방법론, 소프트웨어 척도 (소프트웨어 규모, 개발노력, 개발기간, 팀 규모), 분석과 설계 방법론, 소프트웨어 시험 및 품질보증, 소프트웨어 신뢰성, 신경망, 뉴로-퍼지