소프트웨어 개발팀 규모 추정 모델
(A Model for Estimation Sftware Development Team Size)

이 상 운 †
(Sang-Un Lee)

요 약 소프트웨어 개발 초기에 개발비용, 소요인력과 기간을 추정하는 것은 소프트웨어공학 분야에서 어렵고도 중요한 문제이다. 이 정보들은 소프트웨어 요구사항 명세서로부터 측정된 소프트웨어 규모인 기능점수를 이용하여 추정한다. 측정된 소프트웨어 규모를 개발하기 위해서는 개발팀을 몇 명으로 구성할 것인가가 문제로 제기된다. 본 논문은 소프트웨어 개발팀의 규모를 추정할 수 있는 모델을 제시한다. 모델을 유도하기 위해 301개 소프트웨어 프로젝트들이 사용되었다. 먼저, 통계적 알고리즘 모델인 회귀모델을 연구하였다. 다양한 데이타 변환과 회귀분석 결과 좋은 성능의 모델을 얻지 못하였다. 따라서, 비알고리즘 모델인 RBF망을 적용하여 잔차가 랜덤하게 분포하고 우수한 성능을 가진 모델을 제안하였다. 본 모델은 소프트웨어 개발에 필요한 개발팀 규모에 대한 기준을 제공함으로써 인력관리 정보로 활용할 수 있다.

키워드 : 개발팀 규모, 개발 기간, 소프트웨어 규모, RBF 망, 데이타 변환, 회귀분석

Abstract Estimation of development cost, effort and time is difficult and a key problem of software engineering in the early stage of software development. These are estimated by using the function point which is measured from a requirement specification. However, it is often a serious question of the staffing level required for the software development. The purpose of this paper is to show us the model which can be used to estimate a size of development team. Three hundred one software projects have been analyzed and studied for the model. First, an analysis was conducted for statistical algorithmic model. After various data transformation and regression analysis, it was concluded that no good model was available. Therefore, non-algorithmic model was suggested for analysis, which has random distribution of residuals and makes good performance using RBF (Radial Basis Function) network. Since the model provides a standard to determine the required size of development team, it can be used as management information.

Key words : Development Team Size, Development Duration, Software Size, Radial Basis Function Network, Data Transformation, Regression Analysis

1. 서 론

최근 들어, 소프트웨어 개발일정(기간)과 비용 추정 결과는 크게 부정확하고, 불충분한 품질을 갖고 있으며, 개발 생산성은 사용자가 요구하는 수준보다 매우 느리게 향상되고 있다. Arthur[1]는 이러한 소프트웨어 개발 실상을 소프트웨어 위기(Software Crisis)라고 하였다. 또한 대형 소프트웨어 프로젝트의 1 퍼센트만이 계획된 기간과 예상 비용한도 내에서 고객을 만족시키며 완료되었으며, 대부분의 프로젝트들은 1년 이상의 일정이 지연되고 초기 예상 비용의 2배 정도가 초과되었다[2]. 이와 같은 이유로 인해, 프로젝트 관리 측면에서 소프트웨어 개발 및 유지보수 비용을 줄이고자 체계적인 연구가 수행되고 있으며, 소프트웨어 비용산정, 소요 인력 및 개발일정을 추정하는 모델을 개발하는 계기가 되었다. 계획단계에서 보다 정확한 추정은 프로젝트를 관리할 때 발생하는 다양한 의사결정, 소요 예산 및 개발인원 할당과 계약체결 여부에 신뢰할 만한 정보를 제공한다.

소프트웨어 개발에 소요되는 비용, 인력과 개발기간을 추정하기 위해서는 소프트웨어 규모를 측정해야만 한다. 사용자가 제기한 요구사항 명세서(Requirement Specification)로부터 측정된 소프트웨어 규모(Size)를

† 정 회 원 : 국방품질관리소
sangun_lee@hanmail.net
논문접수 : 2001년 10월 5일
심사완료 : 2002년 9월 16일

이용해 그림 1과 같이 개발에 소요될 노력(E, Effort)과 개발기간(D, Duration)을 추정함과 더불어 개발 소요 비용(Cost)도 추정하였다.

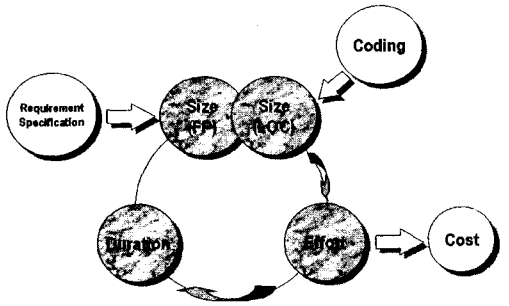

그림 1 소프트웨어 개발인력 및 개발기간 추정 과정

소프트웨어 규모를 측정하기 위한 측도로 가장 많이 사용되고 있는 방법들 중에 LOC(Line Of Code)와 FPA(Function Point Analysis)가 있다. LOC 측도는 언어에 종속되어 있으며, 요구분석 또는 설계단계에서 정확한 추정이 어렵고 코딩이 완료된 시점에서 정확한 측정이 가능한 단점이 있다. 이에 반해, FPA는 사용자에게 양도될 시스템의 기능에 기초하여 소프트웨어 시스템의 규모와 복잡도를 정량화하는 방법으로 소프트웨어 프로젝트를 개발하기 위해 사용되는 언어 또는 도구와 독립적이며, 개발 생명주기의 초기단계인 요구분석 단계에서 측정 가능한 장점이 있어 LOC를 사용할 때의 문제점을 극복할 수 있는 접근법이다[3]. 사용자의 요구사항 명세서에 따라 기능점수(FP, Function Point)를 계산하는 방법에 대해 다양한 연구가 수행되었으며, 정형화된 방법은 IFPUG(International Function Point Users Group) Function Point Counting Practices Manual, Release 4.1[4]에 나타난다.

측정된 기능점수 FP 값을 이용해 소프트웨어 개발에 투입될 노력 E을 추정하는 연구로서 알고리즘 모델인 통계적 분석 방법으로부터 유도된 회귀분석 모델로는 Albrecht[5,6], Albrecht et al.[7], Kemerer[8,9], Low et al.[10], Matson et al.[3] 등이 있으며, 소프트웨어 프로젝트의 개발기간 D를 예측하기 위해 Bailey-Basili('81), Boehm의 COCOMO[11,12], Jones [13], Oligny et al.[14,15] 등 많은 모델들이 제안되었다. 또한, 비알고리즘 모델로 신경망을 적용하여 개발노력을 추정한 모델로는 Gray et al.[16], Hughes[17], Jorgenson[18], Serluca[19], Srinivasan et al.[20], Venkatachalam[21]과 Wittig et al.[22] 등이 있다.

많은 조직에서 제품 출시 시점 또는 구현의 문제로 인해 프로젝트를 시작하기 전에 프로젝트의 개발기간 D가 확정되는 경우가 빈번히 발생한다. 개발 대상 소프트웨어의 규모와 개발기간이 확정된 경우, 개발에 투입될 인력이 얼마나 될 것인가를 관리자 입장에서는 필수적으로 알고 있어야 하는 정보이다. 소프트웨어 개발분야에서의 프로젝트 수행 팀 규모에 관한 연구로서, 정보시스템 개발에 관한 팀 규모 연구는 Samprevivo[23], LOC(Line Of Code)를 대상으로 연구한 사례는 Putnam et al.[24]이 있으며, 기능점수(FP, Function Point)를 대상으로 한 연구사례는 ISBSG (International Software Benchmarking Standards Group) Benchmark Release 6[25]가 있다. 그러나 소프트웨어 개발팀의 적절한 규모에 대한 체계적인 연구는 수행되지 않고 있다. 따라서, 본 논문은 소프트웨어 규모인 기능점수 FP에 대해 개발팀의 규모(TS, Team Size)를 추정하는 모델을 제시한다.

2장에서는 소프트웨어 개발팀 규모 추정에 관한 관련 연구 및 연구동기를, 3장에서는 소프트웨어 개발팀 규모 추정을 위해 알고리즘 기법인 회귀모델과 비알고리즘 기법인 RBF 망을 적용하여 알고리즘 방법의 문제점을 해결하고 개발팀의 규모 TS를 최적으로 추정할 수 있는 모델을 제시한다. 4장에서는 제안된 모델과 기존 연구 결과를 비교 분석하여 제안된 모델의 성능을 평가해 본다.

2. 관련 연구 및 연구동기

프로젝트에 대한 가장 실제적이고 생산적인 팀 규모를 선택하는데 지침으로 사용될 수 있는 정보를 제공하기 위해 프로젝트 규모에 따른 개발팀 규모를 추정한 연구를 고찰해보자. Samprevivo[23]는 정보시스템 개발에 대한 팀 규모를 결정함에 있어서 팀의 규모는 팀 구조가 가장 중요한 관점이며, 그룹의 성취도와 문제 해결 능력에 있어서 팀 요원 개개인의 만족도에 영향을 받음을 밝혔다. 팀 규모와 개별 요원의 불만족 관계에 대해 연구한 결과 팀의 규모가 증가함에 따라 개별 요원의 불만족이 증가하였다. 즉, 작은 그룹(5 - 7명)에서 개별요원은 가장 큰 만족을 느끼고 있으며, 큰 그룹(12 - 15명)에서는 가장 작은 만족을 느낌을 제시하였다. Putnam et al.[24]은 다음과 같은 기준을 제시하였다 : (1) 작은 규모로 적절히 선택된 팀은 성공적인 프로젝트를 위해 필수적이다 ; (2)

실제로 큰 규모의 개발팀이 개발한 프로젝트보다 작은 팀이 적은 노력으로 보다 빨리 프로젝트를 종료한다 ; (3) 20 또는 보다 많은 팀 요원으로 개발된 프로젝트는 5명 또는 그 이하의 요원으로 개발된 프로젝트보다 많은 노력이 필요하다. 이는 동일한 SLOC(Source Line Of Code) 프로젝트 비교로부터 유도되었다. ISBSG Benchmark Release 6[25]는 다음과 같은 기준을 제시하였다 : (1) 적은 팀은 프로젝트 규모와 팀 규모간에 단순 관계를 갖고 있다 ; (2) 2-3명의 팀은 팀 요원 당 가장 큰 규모의 프로젝트를 개발한다 ; (3) 항상 가장 적은 실제 팀 규모를 목표로 삼는다 ; (4) 팀 규모가 5명을 초과하면 생산성은 감소하며, 적은 팀이 보다 생산적이다. 5 또는 그 이상의 팀 규모로 개발된 프로젝트는 보다 적은 팀으로 개발된 프로젝트보다 기능점수 1점을 개발하는데 특히 많은 노력이 소요된다. ISBSG Benchmark Release 6[25]의 연구결과를 보다 자세히 고찰해 보자. ISBSG Benchmark Release 6[25]는 기능점수 *FP*와 개발팀 규모 *TS*를 알고 있는 244개 프로젝트를 대상으로 표 1과 표 2의 결과를 얻었다.

평균적인 개발팀 규모 *TS*는 일반적으로 프로젝트 규모 *FP*가 커짐에 따라 증가하였다. 그러나 Mean과 Median 값은 100 *FP* - 1400 *FP*에서 4 - 6명이었으며, 그 이상의 *FP*에서는 실질적으로 개발팀 규모 *TS*가 증가하였다. 또한, 적은 개발팀 규모 *TS*에 대해서는 프로젝트 규모 *FP*와 개발팀 규모 *TS*사이에 단순한 관계를 가지고 있으며, 2 - 3명의 개발팀 요원으로 구성했을 경우, 개발팀 요원 1명이 개발한 기능점

표 1 프로젝트 규모 *FP*에 따른 개발팀 규모 *TS*

기능점수(*FP*)	표본 수	개발팀 규모(*TS*)			
		Min	Max	Median	Mean
1 - 100	34	1	11	3	3.9
101 - 200	41	1	53	4	5.8
201 - 300	46	1	14	4	5.0
301 - 400	29	1	21	5	6.2
401 - 500	12	2	8	5	5.1
501 - 600	10	4	17	7	8.4
601 - 700	4	3	20	4	7.5
701 - 800	7	3	12	4	5.0
801 - 900	9	2	7	3	4.4
901 - 1,000	4	4	17	8	9.0
1,001 - 1,100	4	3	6	4	4.3
1,101 - 1,200	4	2	12	4	5.5
1,201 - 1,400	9	2	16	4	6.9
1,401 - 1,600	7	3	65	7	15.4
1,601 - 1,800	4	5	8	7	6.8
1,801 - 2,000	3	3	25	11	13.0
2,001 - 3,000	8	3	26	11	12.5
3,001 - 4,000	3	8	15	10	11.0
4,001 - 5,000	3	12	50	24	28.7
5,001 이상	3	8	10	8	8.7

표 2 개발팀 규모 *TS*에 따른 프로젝트 규모 *FP*

개발팀 규모 (*TS*)	표본 수	기능점수(*FP*)			
		Min	Max	Median	Mean
1	19	25	350	82	115
2	33	11	1336	236	327
3	35	39	2145	297	565
4	30	39	1437	329	521
5	24	32	2014	291	499
6	19	33	1093	200	309
7	15	54	1694	499	679
8	15	204	9803	445	1930
9	7	154	1203	303	424
10	12	89	5789	234	1089
11	11	60	1882	374	620
12	5	231	4562	769	1443
13 이상	19	101	4913	1457	1621

수 *FP*(즉, 생산성)가 가장 크게 나타났다.

*FP*가 3,000 이상인 프로젝트는 프로젝트 규모를 표현할 수 있는 대표적인 표본이 나타나지 않는다. 경험적으로 볼 때, 팀 규모가 30 - 50이면 프로젝트의 인도율(*PDR*, Project Delivery Rate) 즉, 기능점수 *FP* 1개를 개발하는데 소요되는 노력(시간)은 20 - 40 시간이 소요되었다. 이는 팀 규모가 증가하면 생산성은 감소하며 인도기간도 지연됨을 의미한다. 프로젝트의 *FP*가 2,000 이상인 프로젝트에 대해 개발팀 규모 관리 요소와 실질적으로 프로젝트의 위험도가 증가한다. 따라서, 일반적으로 표 3의 일반화된 결과를 제시하였다.

표 3 개발팀 규모 선정 결과

기능점수(*FP*)	개발팀 규모(*TS*)		
	Min	Max	Average
1 - 300	1	6	3 - 6
300 - 500	2	10	5
500 - 1,000	4	12	6
1,000 - 1,500	5	15	5 - 7
1,500 이상	7	25	10

ISBSG Release 6[25]에서 제시한 표 3의 결과를 이용하여 개발팀 규모를 추정하여 실측치와 비교한 결과는 그림 2에 제시되어 있다. 그림에서 *FP*가 1-300에서는 개발팀 규모를 4.5로, 1,000 - 1,500 범위에서는 6을 적용하였다. 회귀분석 결과 모델은 12.86%의 정확도로 개발팀 규모를 추정할 수 있었다. 제시된 개발팀 규모 *TS*는 평균값을 적용한 예로, 이론적인 근거에 바탕을 두고있지 못하며, 12.86%의 정확도는 좋지 않은 결과로 일반적인 회귀분석 모델로는 채택될 수 없다.

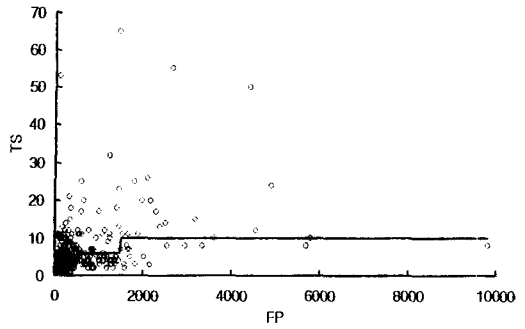

그림 2 ISBSG Release 6[25] 기준에 의한 개발팀 규모 측정

따라서, 이론적인 근거에 바탕을 두고 있으며, 추정 성능도 향상된 소프트웨어 개발팀 규모 TS를 추정할 수 있는 모델 제시가 필요하다.

3. 개발팀 규모 추정 모델

3.1 회귀모델

회귀분석의 첫 단계는 적절한 모델을 가정하고 회귀방정식의 추정을 시작하는 것이다. 그러나 실제로는 적절한 회귀모델이 무엇인지 모를 때가 대부분으로 적절한 회귀모델 선택이 가장 중요한 문제이다. 일반적으로 적절한 회귀모델 선택방법은 다음 절차를 따른다.

Step 1. 두 변수 x, y에 대한 자료 $(x_i, y_i), i=1,2,\cdots,n$을 수집하여 산점도를 그린다.

Step 2. 산점도로부터 어떠한 회귀모델이 적절한지를 선택한다.

Step 3. 선택된 회귀모델을 자료에 적합시켜 회귀 방정식을 구한 후, 이 회귀모델이 올바른 선택되었는지를 다음 기준에 의해 판단한다.

a. 분산분석표에 의한 유의성 검정 결과 유의하지 않으면 선택된 회귀모델은 옳지 않다.

b. 결정계수 R^2의 값이 충분히 크지 않으면 선택된 회귀모델이 옳지 않다.

c. 잔차 분석을 통해 선택된 회귀모델이 옳은지를 판단한다.

Step 4. Step 3의 판단기준에 의해 선택된 회귀모델이 옳지 않으면 Step 2로 복귀한다.

회귀분석의 경우 회귀직선에 의해 종속변수가 설명되는 정도를 결정계수(Coefficient of determination, R^2)라 한다. 즉, 종속변수의 값은 독립변수에 의해 결정되는 부분과 미지의 오차의 합으로 나타나며, 총 변동을 설명하는데 있어서 회귀직선에 의해 설명되는 변동 비율이 R^2이다. 따라서, $R^2(0 \le R^2 \le 1)$이 0에 가까우면 추정된 회귀직선은 쓸모가 없으며, 값이 클수록 쓸모 있는 회귀직선이 된다. 잔차(Residual)는 실제 값과 추정된 값과의 차이로, 잔차분석은 단순회귀모델에서 설정한 등분산성, 독립성, 정규성과 직선관계의 가정이 옳은가를 검토할 때 가장 많이 사용되는 방법이다. 따라서, 잔차가 어떤 일정한 형태를 취하지 않고 랜덤하게 분산되어 있는 경우가 좋은 모델이 된다.

소프트웨어 규모인 기능점수 FP에 따른 개발팀 규모 TS를 회귀분석한 결과 선정된 모델은 그림 3과 같이 가장 좋은 성능을 가진 누승 모델의 성능이 19.40%로 적합한 회귀모델로 선정이 불가능하다. 따라서, 일반적인 회귀분석 방법을 적용하기가 불가능함을 알 수 있다.

일반적으로 선형(단순) 회귀모델을 이용할 경우, 종속변수(개발팀 규모 TS)를 설명하는데 독립변수(기능점수 FP)의 일차함수로서 충분하다는 가정이 옳다고 생각하고 회귀분석을 수행하는 과정을 다룬다. 실제로 추정하고자하는 개발팀 규모 TS와 소프트웨어 규모인 기능점수 FP 관계는 그림 3과 같이 선형보다는 비선형 형태의 관계를 갖는 경우가 대부분이며, 자료의 분석에 있어서 단순회귀분석처럼 간단히 처리되지 않는다. 이 경우, 간단한 변수 변환(Variable Transformation)을 통해 직선관계로 변형시킬 수 있으며, 직선회귀 방법을 사용하여 분석한 후 변환되기 이전의 변수간의 관계를 규명하는 경우도 있다. 이 방법에 의해, 로그, 자연로그 등 다양한 변수변환 방법을 실험한 결과 그림 3의 경우보다 뚜렷한 성능향상을 보인 모델을 찾는데 실패하였다.

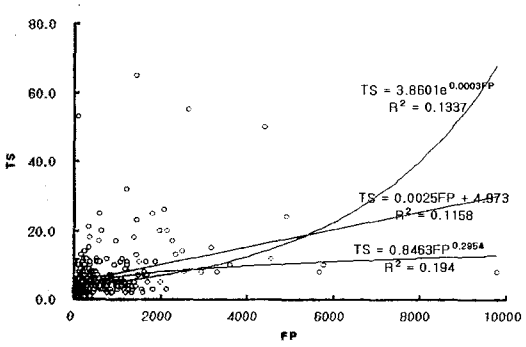

그림 3 개발팀 규모 TS와 기능점수 FP의 관계

그림 4는 로그 데이타 변환 후 회귀분석 결과 얻은 모델로 20% 이하의 성능으로 개발팀의 규모를 추정할 수 있음을 보여주고 있다. 따라서, 알고리즘적 모델인 통계적 회귀분석 방법으로는 소프트웨어 개발팀 규모 *TS*를 추정하는 적합한 모델을 유도할 수 없으며, 비알고리즘 방법을 적용하여 문제를 해결하는 것이 필요하다.

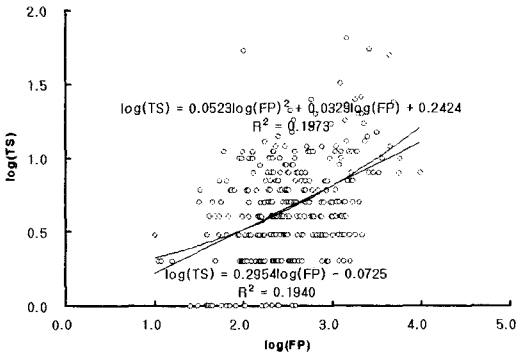

그림 4 log 데이타 변환에 따른 개발팀 규모 추정

3.2 RBF 망 모델

3.2.1 RBF 망 소개

알고리즘적 방법인 통계적 분석기법으로부터 유도되는 회귀분석 모델로는 현대의 복잡한 문제를 해결하기가 어려운 경우가 대부분이다. 따라서, 이의 대안으로 비알고리즘 방법들이 제안되었으며, 이들 중에 신경망, 퍼지 또는 뉴로퍼지 등이 있다. 신경망은 음성인식, 적응제어 등 분야에서 많은 어려운 문제들을 해결하는데 성공적으로 적용되고 있으나 소프트웨어 개발노력이나 비용 등을 추정하는 소프트웨어공학분야에는 최근들어 적용되고 있다. 신경망 중에서 가장 일반적으로 적용되는 망이 역전파(Backpropagation) 학습 알고리즘을 적용한 FFN(Feedforward Network)과 LMS(Least Mean Square) 학습 알고리즘을 적용하는 RBF(Radial Basis Function) 망이다. 역전파 학습알고리즘을 사용하는 FFN은 은닉 뉴런 수를 자동적으로 결정하는 방법이 상용화되지 않았으며, 랜덤한 초기값에 의해 훈련을 수행할 때마다 망의 출력이 다르게 나타나 여러 번의 시행 결과를 평균하여 출력값을 결정하는 방법을 이용하는 단점이 있다. 이에 비해, RBF 망은 최적의 은닉 뉴런 수(기저함수 개수)를 결정함에 있어 주어진 문제에 적합하도록 은닉 뉴런 수를 0개로부터 시작하여 점진적으로 추가시키면서 주어진 문제에 적합한 뉴런 수를 찾는 방법을 사용할 수 있으며, LMS(Least Mean Square) 알고리즘을 사용하여 훈련을 수행함으로써, 망의 출력이 항상 동일한 결과를 얻을 수 있어 FFN 적용에 비해 장점을 갖고 있다. 또한, RBF 망은 전형적인 다층 FFN 보다 빠른 수렴속도, 보다 적은 외삽법 오차와 높은 신뢰성으로 인해 공학분야에서 점점 더 관심의 대상이 되고 있다. 따라서, 본 논문은 신경망 중에서 FFN 대신 RBF 망을 적용하여 소프트웨어 개발팀 규모를 추정하는 모델을 개발하고자 한다.

벡터공간 V의 벡터집합을 S라하고, S에 있는 모든 벡터가 서로 1차 독립이며, V의 모든 벡터는 S에 있는 모든 벡터의 1차 결합(선형결합)으로 표현될 수 있을 때 S를 V의 기저함수(Basis Function)라 한다. 따라서, RBF 망은 주어진 입력들을 분류할 수 있는 군집(Clustering)을 결정하는 기저함수를 찾는 것으로, 기저함수만 있으면 입력벡터 공간의 모든 점들을 기저함수의 선형결합으로 표현할 수 있다는 개념에서 유래되었다. RBF 망은 그림 5의 2개 층(은닉층과 출력층)으로 구성되어 있다. 은닉층 뉴런은 입력의 가중합을 취하지 않으며, 은닉 뉴런의 작동함수(Activation Function)는 가우시안(Gaussian) 함수(e^{-x^2})와 같은 방사 대칭 기저함수(Radially Symmetric Basis Function)를 사용하여 FFN과 다른 특징을 갖고 있다. 그림 6과 같이 각 은닉층 뉴런의 출력(a_1)은 기저함수의 중심과 망의 입력간 거리에 의해 결정되는 기저함수로 표현되며, 기저함수에 근접된 입력에만 응답함으로써, 지역적으로 응답하는 영역을 가진다. 이것은 전역 응답을 생성하는 시그모이드 작동함수를 가진 표준 다층 신경망인 FFN(Feed Forward Network)과 상반되는 역할을 수행한다. 또한, 출력층은 선형 작동함수를 가지며 선형결합으로 은닉층 출력의 가중합을 생성한다.

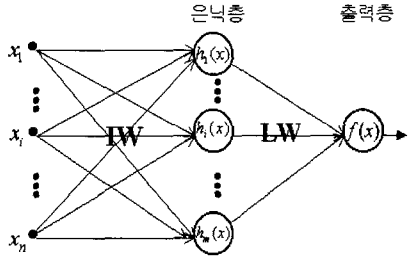

그림 5 RBF 망

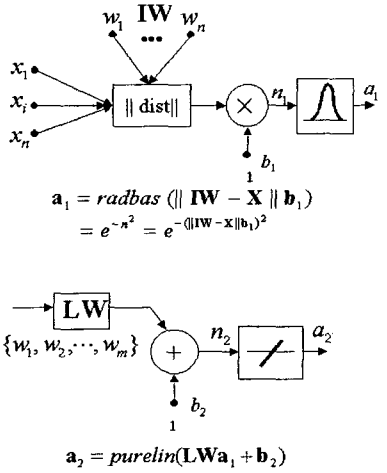

그림 6 은닉층과 출력층 출력

3.2.2 RBF 망 설계

RBF 망은 FFN 설계시 제기되는 몇 개의 은닉층과 은닉층에 몇 개의 은닉 뉴런을 사용할 것인가의 문제는 발생되지 않는다. 단지 1개의 은닉층을 사용하고 은닉층의 뉴런 개수는 최악의 경우 입력 데이타의 최대 개수까지만 되며, 최적의 개수는 망의 훈련과정에서 자동적으로 결정된다[26].

RBF 망 설계시 중요한 사항은 입력과 출력을 결정하는 것이다. 이는 어떤 데이타를 입력으로 하여 어떤 데이타를 얻을 것인가에 의존하며, 훈련을 시작하기 이전에 결정해야 한다. 소프트웨어 개발팀 규모 *TS*를 추정하는 목적을 달성하기 위해, 소프트웨어 개발팀 규모와 관련되어 수집된 변량(Variable) 들을 입력으로 선택할 수 있다. 소프트웨어 규모인 기능점수 *FP*, 개발에 소요될 노력 *E*, 사용자가 요구하는 확정된 개발기간 *D* 자료가 소프트웨어 개발 초기에 수집되었다고 가정한다. 이들 데이타로부터 *PDR*(Project Delivery Rate) 측도가 측정되었다고 하자. ISBSG Benchmark Release 6[25]는 소프트웨어 기능을 사용자에게 인도하는 비율인 프로젝트 인도율 *PDR*을 개발노력 *E*의 계수로 측정하였다. 즉, 프로젝트 개발노력 *E* 전체에 대해 인도된 소프트웨어의 기능 규모 *FP*로 정의하였으며, *E*/*FP*로 계산된다. 이는 기능점수 *FP* 1개를 개발하는데 소요되는 개발노력(시간)이다. 그림 7과 같이 *PDR*을 고려하지 않고 모델의 입력으로 *FP*만 이용하는 경우, *FP*, *E*와 *D*를 이용하는 경우, *FP*와 *D*를 이용하는 경우와 그림 8과 같이 *PDR*을 고려하여 *FP*와 *PDR*을 이용하는 경우와 *FP*, *PDR*과 *D*를 이용하는 경우에 대해 소프트웨어 개발팀의 규모 *TS*를 추정하는 망을 설계한다.

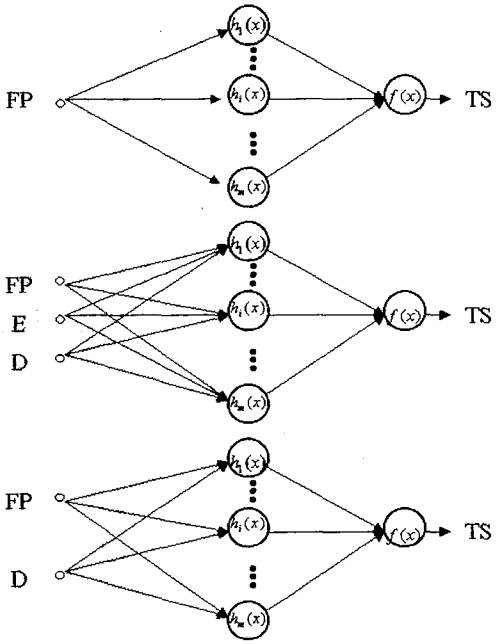

그림 7 *PDR*을 고려하지 않은 RBF 망 입·출력 결정

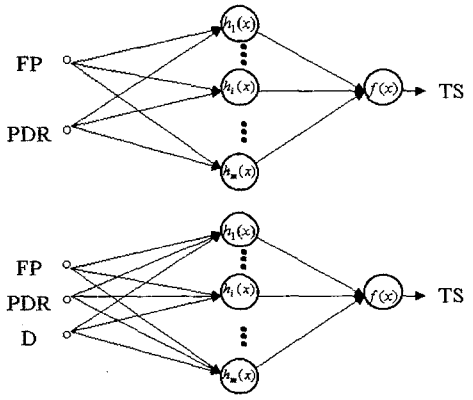

그림 8 *PDR*을 고려한 RBF망 입·출력 결정

기능점수 *FP*를 입력으로 하는 모델을 RBF_F, *FP*, *E*와 *D*를 입력으로 하는 모델을 RBF_{FED}, *FP*와 *D*를 입력으로 하는 모델을 RBF_{FD}라 하자. 또한, *FP*와 *PDR*을 입력으로 하는 모델을 RBF_{FP}, *FP*, *PDR*과 *D*

를 입력으로 하는 모델을 RBF_{FPD}라 하자.

RBF 망을 적용하는데 있어서 다음으로 해결해야 할 사항은 기저함수의 폭을 결정하는 것이다. 기저함수의 폭은 입력 영역에서 각 기저함수가 표현할 수 있는 영역(지역 정보 표현 범위)을 결정하는 것으로 일반적으로 0.1과 2 사이의 값을 취하고 있으며[27], 본 논문에서는 일반적으로 많이 적용되고 있는 1.0을 취한다.

망의 훈련을 종료하는 기준은 훈련과정에서 망의 출력이 설정된 SSE(Sum Square Error) 이하의 결과를 얻거나 기저함수 개수가 입력 데이타의 최대 개수만큼 되는 기준 중 어느 하나를 만족할 때까지 기저함수를 하나씩 추가하면서 훈련을 수행하는 방법[26]을 이용한다.

4. 실험 및 결과 분석

제안된 5개의 모델 각각에 대해 ISBSG Benchmark Release 6[25]의 301개 프로젝트에 대해 개발팀의 규모를 추정하였다. 분산분석 결과는 표 4에 제시되어 있으며, 종합 결과는 표 5와 같으며, 모델을 평가하기 위해 R^2와 MMRE를 계산하였다.

표 4 분산분석

(1) RBF_F 모델

	자유도	제곱합	제곱평균	F비	유의한 F
회귀	1	223975.2920	223975.2920	13428.6895	1.6237E-250
잔차	299	4986.9805	16.6789		
계	300	228962.2724			

(2) RBF_{FED} 모델

	자유도	제곱합	제곱평균	F비	유의한 F
회귀	1	224248.5130	224248.5130	14224.3799	3.5641E-254
잔차	299	4713.7595	15.7651		
계	300	228962.2724			

(3) RBF_{FD} 모델

	자유도	제곱합	제곱평균	F비	유의한 F
회귀	1	226770.1123	226770.1123	30930.3418	6.9425E-304
잔차	299	2192.1602	7.3316		
계	300	228962.2724			

(4) RBF_{FP} 모델

	자유도	제곱합	제곱평균	F비	유의한 F
회귀	1	228862.5344	228862.5344	686096.1997	0.0000
잔차	299	99.7381	0.3356		
계	300	228962.2724			

(5) RBF_{FPD} 모델

	자유도	제곱합	제곱평균	F비	유의한 F
회귀	1	228863.6933	228863.6933	694165.7353	0.0000
잔차	299	98.5781	0.3297		
계	300	228962.2724			

표 5 모델 성능 종합

모델	기저함수 개수	훈련 종료 기준(SSE)	모델 성능 (R^2)	표준 오차	정확도 MMRE
RBF_F	17	5,000	0.9782	4.0840	90.4786
RBF_{FED}	10	5,000	0.9794	3.9705	90.4170
RBF_{FD}	33	2,200	0.9904	2.7077	71.3252
RBF_{FP}	188	100	0.9996	0.5776	14.8179
RBF_{FPD}	177	100	0.9996	0.5742	17.1891

모델 정확도(Accuracy)를 평가하는 측도로 MMRE (Mean Magnitude of Relative Error)가 있다. 이 측도는 다음과 같이 측정된다. RE(Relative Error) $= \frac{\text{추정치} - \text{실측치}}{\text{실측치}}$, MRE(Magnitude of RE) = $|RE|$, MMRE(Mean MRE) = $100/n * \sum_{i=1}^{n} MRE$. 유의성 검정 ($F$비 ≥ 유의한 F) 결과 제안된 5개 모델 모두 유의하여 올바로 선택된 모델임을 알 수 있다. 모델의 성능 비교 결과 PDR을 고려한 RBF_{FP}와 RBF_{FPD} 모델들이 가장 좋은 99.96%의 성능으로 개발팀의 규모를 추정할 수 있음을 알 수 있다. 또한, RBF_{FP} 모델이 가장 좋은 정확도(최소 MMRE)로 개발팀의 규모를 추정할 수 있다.

RBF_F 모델의 훈련 결과는 그림 9에 제시하였다.

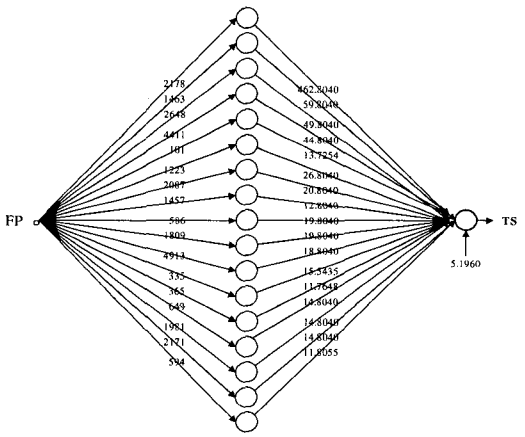

그림 9 소프트웨어 개발팀 규모 추정 RBF_F 모델

만약, 소프트웨어 개발팀의 규모를 추정하기 위해 RBF_F 모델을 이용하고자 한다면, 그림 9의 가중치를 할당하고 표 5의 훈련 종료기준을 적용한 RBF망을 구성하고 개발하고자 하는 소프트웨어의 규모 FP를 입력하면 원하는 개발팀의 규모 TS를 얻을 수 있다.

RBF_{FP} 모델의 잔차를 분석한 결과는 그림 10에 제시하였다. 잔차분석 결과, 잔차들이 랜덤하게 분포되어 좋음 모델임이 판명되었다. 다른 4개 모델들의 잔차도 유사한 패턴을 갖고 있다.

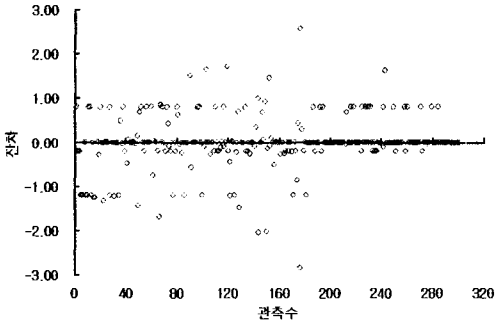

그림 10 RBF_{FP} 모델 잔차

제안된 모델의 성능으로 소프트웨어 규모인 기능점수 FP에 따른 개발팀의 규모를 실측치와 비교한 결과는 그림 11과 그림 12에 제시하였다. 그림 11은 RBF_F 모델의 성능을 나타내고 있으며, RBF_{FED}와 RBF_{FD} 모델의 성능도 유사한 경향을 나타내고 있다. 그림 12는 RBF_{FP} 모델의 성능을 나타내고 있으며, RBF_{FPD} 모델도 유사한 경향을 나타내고 있다.

그림에서 개발팀의 규모를 추정한 결과를 비교하면 PDR을 고려한 모델이 PDR을 고려하지 않은 모델들보다 월등한 성능 향상으로 개발팀 규모를 추정할 수 있음을 알 수 있다. 모델의 성능 R^2, 정확도 MMRE,

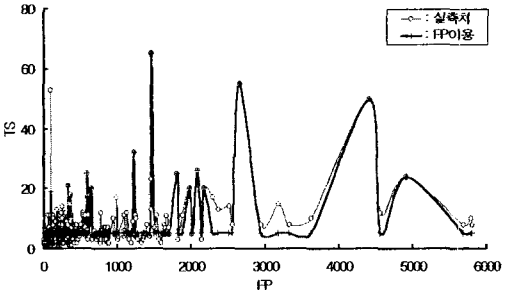

그림 11 RBF_F 모델의 개발팀 규모 추정

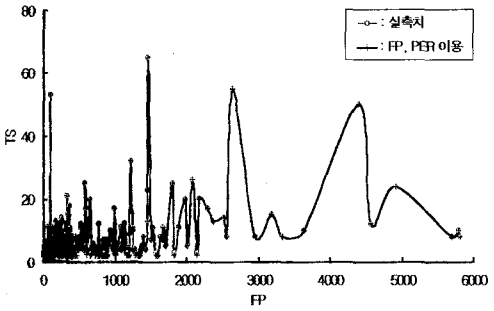

그림 12 RBF_{FP} 모델의 개발팀 규모 추정

잔차분석 등을 평가한 결과, 소프트웨어 규모인 기능점수 FP와 인도율 PDR을 이용하는 RBF_{FP} 모델이 가장 좋은 모델로 선정되었으며, 99.96%의 성능으로 개발팀의 규모를 거의 완벽하게 추정할 수 있다. 이 모델은 개발기간 D 정보를 이용하지 않는다. 따라서, 개발될 소프트웨어의 규모와 개발팀의 프로젝트 인도율 능력만 알고 있으며 개발팀의 규모를 추정할 수 있다

5. 결 론

소프트웨어 규모와 개발기간이 주어졌을 때, 프로젝트를 원활히 개발하기 위해 개발팀 규모를 결정하는 것은 프로젝트 관리자에게 필수적으로 요구되는 정보이다. 그러나 개발팀 규모를 추정하기 위해 제안된 모델이 없는 실정이다.

본 논문은 먼저 회귀분석을 통해 적절한 모델을 찾고자 하였으나 소프트웨어 규모와 개발팀 규모 간에 비선형적인 관계로 인해 모델 개발에 실패하였다. 이어서, 비선형적인 관계를 해결하는데 이용되는 변수변환 방법을 이용해 회귀모델을 찾고자 하였으나 20% 이하의 성능만을 나타내 좋은 모델을 개발하지 못했다. 결론적으로, 알고리즘적 모델인 통계적 회귀모델로는 개발팀의 규모를 정확히 추정할 수 없음을 보였다. 따라서, 비알고리즘적 모델을 이용해 개발팀의 규모를 추정하고자 하였다. 신경망의 일종인 RBF 망을 적용하였으며, 제안된 모델은 ISBSG Benchmark Release 6[25]의 최근 다양한 개발환경과 방법론으로 개발된 789개 프로젝트 중 소프트웨어 규모, 개발기간과 개발팀 규모가 모두 명시된 301개 프로젝트들로부터 유도하였다.

5개의 RBF 망 모델들이 제안되었으며, 5개 모델들 중에 모델의 성능, 정확도, 잔차 분석 등을 평가한 결과 개발기간에 무관하게 개발될 소프트웨어의 규모와

개발팀의 프로젝트 인도율 능력에만 영향을 받는 모델이 가장 좋은 성능을 나타내었다. 제안된 모델은 99.96%의 성능으로 개발팀 규모를 추정할 수 있는 능력을 갖고 있으며, 잔차들도 랜덤하게 분포하여 좋은 모델임이 증명되었다.

따라서, 개발팀의 프로젝트 인도율 능력은 경험적으로 알고 있다고 할 때, 개발될 소프트웨어 규모인 기능점수 *FP* 만 측정하면 본 모델을 이용해 99.96%의 성능으로 개발에 투입해야 할 개발팀 규모를 판단할 수 있다. 본 모델은 소프트웨어 규모가 주어졌을 때, 소프트웨어 개발에 필요한 인력을 산출하는 기준 정보로 활용할 수 있다.

참 고 문 헌

[1] Arthur. L. J., "Measuring Programmer Productivity and Software Quality," New York, Jone Wiley, 1985.

[2] Möller, K. H. and Paulish, D. J., Software Metrics-A Practitioner's Guide to Improved Product Development, Chapman & Hall Co., New York, 1993.

[3] Matson, J. E., Barrett, B. E. and Mellichamp, J. M. "Software Development Cost Estimation Using Function Points," IEEE Trans. on Software Eng., Vol.20, No.4, pp. 275-287, 1994.

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

[5] Albrecht, A. J., "Measuring Applications Development Productivity," Proceedings of IBM Application Dev., Joint SHARE/GUIDE Symposium, Monterey, CA., pp. 83-92, 1979.

[6] Albrecht, A. J., "Measuring Application Development Productivity," In Programming Productivity : Issues for the Eighties, C. Jones, ed. Washington, DC : IEEE Computer Society Press, 1981.

[7] Albrecht, A. J. and Gaffney, J. E., "Software Function, Source Line of Code and Development Effort Prediction : A Software Science Validation," IEEE Trans. on Software Eng., Vol. SE-9, No.6, pp. 639-648, 1983.

[8] Kemerer, C. F., "An Empirical Validation of Software Cost Estimation Models," Communication ACM, Vol.30, No.5, pp. 416-429, 1987.

[9] Kemerer, C. F., "Reliability of Functional Point Measurement - A Field Experiment," Communications of ACM, 1993.

[10] Low, G. C. and Jeffery, D. R., "Function Points in the Estimation and Evaluation of the Software Process," IEEE Trans. on Software Eng., Vol. 16, pp. 64-71, 1990.

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

[12] Boehm, B. W., "Software Engineering Economics," IEEE Trans. on Software Eng., Vol.10, No.1, pp. 7-19, 1984.

[13] Jones, C., "Determining Software Schedules," Computer Vol. 28, No. 2, pp. 73-75, 1995.

[14] Oligny, S., Bourque, P. and Abran, A., "An Empirical Assessment of Project Duration Models in Software Engineering," In The Eight European Software Control and Metrics Conference(ESCOM'97), Berlin Germany, 1997.

[15] Oligny, S., Bourque, P., Abran, A. and Fournier, B., "Exploring the Relation Between Effort and Duration in Software Engineering Projects," World Computer Congress 2000, August 21-25, Beijing, China, pp. 175-178, 2000.

[16] Gray, A. R. and MacDonell, S. G., "A Comparison of Alternatives to Regression Analysis as Model Building Techniques to Develop Predictive Equations for Software Metrics," University of Otago, 1996.

[17] Hughes, R. T., "An Evaluation of Machine Learning Techniques for Software Effort Estimation," University of Brighton, 1996.

[18] Jorgenson, M., "Experience with the Accuracy of Software Maintenance Task Effort Prediction Models," IEEE Trans. on Software Engineering, Vol. 21, No. 8, pp. 674-681, 1995.

[19] Serluca, C., "An Investigation Into Software Efffort Estimationusing a Back-propagation Neural Network," M.S. Thesis, Bournemouth University, 1995.

[20] Srinivasan, K. and Fisher, D., "Machine Learning Approaches to Estimating Software Development Effort," IEEE Trans. on Software Engineering, Vol. 21, No. 2, pp. 126-136, 1995.

[21] Venkatachalam, A. R., "Software Cost Estimation Using Artificial Neural Networks," In International Joint Conference on Neural Networks, Nagoya : IEEE, 1993.

[22] Wittig, G. and Finnie, G., "Estimating Software Development Effort with Connectionist Models," Information and Software Technology, Vol. 39, pp. 469-476, 1997.

[23] P. C. Semprevivo, "Teams in Information Systems Development," Yordon, pp. 85, 1980.

[24] L. Putnam and W. Myers, "Selecting the Right Team Size : Small is Beautiful," Cutter Consortium, 1998. 12. "http://www.cutter.com/consortium/research/1998/crb981222.html

[25] ISBSG, "Worldwide Software Development - The Benchmark Release 6," Victoria, Australia International Software Benchmarking Standards Group, 2000.

[26] Broomhead, D. S. and Lowe, D., "Multivariate Functional Interpolation and Adaptive Networks," Complex Systems, Vol. 2, pp. 321-355, 1988.

[27] Chen, S., Cowan, C. F. N. and Grant, P. M., "Orthogonal Least Square Learning for Radial Basis

Function Networks," IEEE Trans. on Neural Networks, Vol. 2, No. 2, pp. 302-309, 1991.

이 상 운

1983년 ~ 1987년 한국항공대학교 항공전자 공학과(학사). 1995년 ~ 1997년 경상대학교 컴퓨터과학과(석사). 1998년 ~ 2001년 경상대학교 컴퓨터과학과(박사). 1992년 ~ 현재 국방품질관리소 항공전자장비 및 소프트웨어 품질보증 담당. 관심분야는 소프트웨어 공학(소프트웨어 시험 및 품질보증, 소프트웨어 신뢰성), 소프트웨어 메트릭스, 신경망, 뉴로-퍼지