축약형 신경망과 휴리스틱 검색에 의한 소프트웨어 공수 예측모델

전 응 섭†

요 약

소프트웨어공수 예측에 관한 전통적인 모델들의 한계점을 극복하기 위해 사례기반과 신경망 그리고 퍼지이론 및 전문가 시스템 등 인공지능 기법을 이용한 연구들이 제시되고 있다. 특히 신경망을 이용한 공수예측 모델들이 예측력에 있어서 전통적인 모델들 보다 우수한 예측결과를 제시하고 있다. 그러나 이들 신경망 모델에 있어서도 고려되어야 할 점은 입력데이터의 노이즈와 모델 설계 및 사용에 있어서 유연성 및 효율성 측면이 제기되고 있다. 본 연구에서는 이러한 기존의 신경망모델의 효율성 향상을 위한 새로운 방안으로 최적의 축약형 모델구조와 이에 관련된 최적 사례들을 사용하기 위한 사례기반 휴리스틱 검색기법을 제시한다. 30여개의 실제로 수행된 프로젝트의 예측결과를 통해 최적 사례기반 축약형 신경망 모델의 결과가 전통적인 COCOMO모델 그리고 기존의 신경망 모델과 비교해서 예측력과 모델의 유연성이 좋아졌음을 알 수 있었다. 따라서 본 연구에서 새롭게 제시한 축약형 신경망 모델과 최적사례기반 접근방법은 급변하는 정보시스템 패러다임하에서도 유용하게 사용될 수 있을 것이다.

Parsimonious Neural Network and Heuristic Search Method for Software Effort Estimation Model

Eung-Sup Jun†

ABSTRACT

A number of attempts to develop methods for measuring software effort have been focused on the area of software engineering and many models have also been suggested to estimate the effort of software projects. Almost all current models use algorithmic or statistical mechanisms, but the existing algorithmic effort estimation models have failed to produce accurate estimates. Furthermore, they are unable to reflect the rapidly changing technical environment of software development such as module reuse, 4GL, CASE tool, etc. In addition, these models do not consider the paradigm shift of software engineering and information systems (i.e., Object Oriented system, Client-Server architecture, Internet/Intranet based system etc.). Thus, a new approach to software effort estimation is needed. After reviewing and analyzing the problems of the current estimation models, we have developed a model and a system architecture that will improve estimation performance. In this paper, we have adopted a neural network model to overcome some drawbacks and to increase estimation performance. We will also address the efficient system architecture and estimation procedure by a similar case-based approach and finally suggest the heuristic search method to find the best estimate of target project through empirical experiments. According to our experiment with the optimally parsimonious neural network model the mean error rate was significantly reduced to 14.3%.

키워드 : 축약형 신경망모델(Parsimonious neural network), 최적사례집단(Similar case-based approach), 사례기반접근법(Reuristic search method), 휴리스틱검색기법(Software effort estimation)

1. INTRODUCTION

Estimation of software effort is one of the most important issues for the effective management of software projects and to make software products more competitive. However, it is very difficult to accurately estimate software effort, as the development of software is a labor-intensive task and an intangible creation process. The development of models for estimating software effort and the assessment of the factors affecting software development have been the focus of many research studies. Although a number of software effort estimation models and methods focusing on algorithmic mechanism have been suggested, there are a lot of complaints and critiques by users. These complaints are mainly concerned with the existing models or methods not having accuracy, fitness, flexibility, or portability. Studies have

† 정 회 원 : 인덕대학 소프트웨어개발과 교수
논문접수 : 2000년 9월 8일, 심사완료 : 2001년 2월 13일

shown that the cost and effort estimates derived from different models seem to have significant variations (Saiedian, Band, and Barney, 1992).

The neural network (NN) model is a more effective conjecture method than the existing mathematical or statistical function model. Methods of improving the estimation, for the most part, have been based on neural network approaches. However, the performance of software effort estimation using a naive neural network alone is not as effective because of all the various data noises included in input factors.

We will focus our research issues on combining the neural network model with a case-based approach and devise an efficient search algorithm in order to improve estimation performance. The purpose of the study is to find the optimal estimation result and to suggest an architecture of a NN model. The goals will be achieved by analyzing the sensitivity between the degree of the similarity of cases, and the number of input factors. To solve this problem, we propose to use the qualitative input factors as criteria of data group. With the data sets that have the same values for certain qualitative input factors, we can eliminate the factors from the model building parsimonious neural network models. We will select the search algorithm to find the best estimation result and suggest the estimation method and procedure for this architecture. We will also evaluate and validate the comparative performance of our model through an empirical test. According to the paired t-test, we could prove that the optimally parsimonious neural network model can significantly reduce the error more effectively than the naive neural network model which uses the all data with the all input factors.

The remaining sections are organized as follows : the performance of the major algorithmic software estimation models and the nave neural network models are reviewed in section 2. The definition of input factors for the naive neural network is described and the performance with a parsimonious neural network model is studied in section 3. The effective and parsimonious type of neural network model with the similar case-based approach is suggested. The system architecture and procedures of software estimation is described in section 4. The performance evaluation of the naive neural network and the parsimonious neural network model with the optimal similarity level through the sensitivity analysis is analyzed in section 5.

2. SOFTWARE EFFORT ESTIMATION MODELS

2.1 MAJOR ALGORITHMIC SOFTWARE ESTIMATION MODELS

There have been several software effort estimation models, such as Boehm's COCOMO model, Putnam's SLIM model, RCA's PRICES model, Jensen's SEER model, Grumman's SOFTCOST model, Albrecht's Function Points etc. Among the current models, COCOMO (Boehm, 1981) and Function Points (Albrecht et. al., 1983) are the predominant software effort estimation models. However, the existing software estimation models have suffered from a number of weaknesses. In using functional algorithmic models such as COCOMO, SLIM and Function Points, managers and estimators of software development projects complain that it is very difficult to apply them directly to a different business environment. They also comment that these models are not flexible enough to be utilized within the current rapidly changing software development environment (i.e., development tools, methodology, new language and any other information technology).

2.2 LIMITATIONS OF EXISTING SOFTWARE ESTIMATION MODELS

There have been many discussions on the accuracy of the existing software effort estimation models. Numerous estimation methods based on algorithmic models are not valid due to a lack of accuracy, according to the research results of Kermerer (1987). He showed that the inaccuracy of algorithmic models are as follow :

- SLIM had 772% error of estimates
- COCOMO had 610% error of estimates
- Function Points had 103% error of estimates

His study indicates that more research is needed to develop a model for software effort estimation for the current software development environment. Venkatachalam (1993) argued that the models could not reflect the recent development technology in the areas of programming languages, hardwares, and methodologies of software engineering, communication and network technology etc..

Through review of several research studies on software effort estimation, we pinpointed the limitations of the existing software effort estimation models. Recently, in order to improve the accuracy of the estimation model, there have been numerous attempts to utilize the artificial intelligence

techniques in software engineering because those techniques are clearly required in the field of software effort estimation. Prietual, et al. (1991) emphasized that a qualitative improvement in estimation could come from expert insight. Vicinanza et al. (1991) suggested that more accurate estimation could be generated by an experienced software development manager. Ramsey (1989) suggested a knowledge-based approach for improving accuracy ; and Wriegly (1987) emphasized that Human judgment remained as the dominant method. Bergeron (1992) surveyed and summarized that the most important estimation methods were the experience, expert, and analogy approaches. Genuchten (1991) got similar results to those stated above.

2.3 NEURAL NETWORK MODELS

Neural Network is very powerful method which is used to find the mapping relationship from a paired data set. It has the following advantages ; robust to non repeatable data or missing data, or strong to data representation and nonlinear effect by hidden layer. Most predicting and decision-making methodological advances have been based on statistical techniques. Artificial neural network is a new challenger for these methodologies (Hill et al., 1994). It has been widely applied to solving many forecasting and decision modeling problems because it can be modeled easily to any type of parametric or non-parametric process and it can transform the input data automatically and optimally into the output (Hiew and Green, 1992). Venkatachalam et al., (1993) recommended the concept of neural network model architecture for software effort estimation. In his research, a back-propagation neural network was constructed with 22 input nodes and two output nodes. The input nodes represented the distinguishing features of software projects, and the output nodes represented the effort required in terms of person-month and the development time required to complete the project. The variables considered for the research were taken from the COCOMO database. In his research, he was unable to compare his neural network model with COCOMO because it was an on-going research.

Srinivasan et al. (1995) compared his neural network model with Kemerer's results obtained from 15 projects. His model used 15 input attributes of product, computer, personnel, and project classified by Boehm. and used the regression trees for learning decisions. He showed that his output was more efficient than that of Boehm's COCOMO.

Jun (1996) designed the neural network model composed of one output (i.e, man-month) and nine inputs that impacted on the software development effort. He suggested that his neural network model had a more efficient performance of estimation compared to other models such as COCOMO and the multi-regression model which were designed by using the same input factors and output. From these models, however, we found that the biggest problem was the data noises, which lead to the large error rate of estimates. And as a result our focus turned to reducing the data noises.

3. NEURAL NETWORK MODEL APPROACH FOR SOFTWARE EFFORT ESTIMATION

3.1 DEFINITION OF INPUT FACTORS FOR NN MODEL

Since there are numerous variables that can influence the amount of effort needed to complete a system development project, we selected the input and output factors after considering the possibilities for the empirical data set collections and the factors which had general features such as Boehm's suggestions in this study.

Our research classified the major factors affecting software effort estimation into four categories : Project features, Product features, Staff features and Technology features. Among these features, Project and Product are extraneous variables which are uncontrollable. Staff and Technology are intrinsic, or decision variables, which are controllable. After surveying the 30 participants who developed and maintained software projects in Korea, we selected and defined the necessary factors by analyzing the importance between the effort and the candidates of input factors for our research model. We also surveyed the priority order of four features. Project features are considered of the first priority order, Product features are the second priority order, and staff features and Technology features are the third and fourth priority order respectively. Consequently, we prioritized the criteria which classified the similar case groups in the following order : project, product, staff and technology. They are defined according to the suggested criteria of Boehm's (1984, 1988) and Bergeron's (1992) researches. In this study, we defined the major input factors more detailed for our NN model by considering the ease and the possibility of data collection as shown in <Table 1>. Here, Used Algorithm is specially noteworthy because of its four detail items ; D/B access, I/O process, Math/Statistics process, and AI based process. They apparently have nominal type of values such as "Yes" or "No", but they are determined to be either "yes" or "no" depending on the density of them that plays an

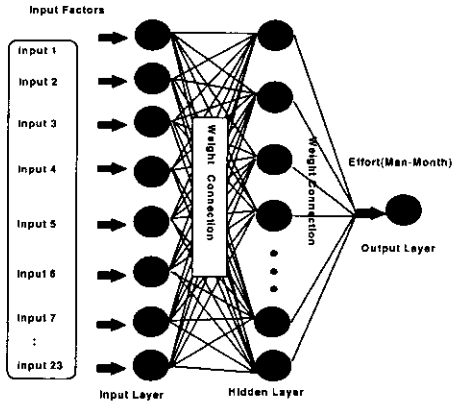
important part in systems. So the quantitative percentage of the relative importance in systems determine values of the Used Algorithm's four items. The values are to be binary depending on the density. Thus, there are eight qualitative input factors used for case grouping criteria as followings ; Duration_Constraint (I1), Development_Type (I4), Processing_Type (I5), Development_tool (I19), Methodology (I20), Language (I21), Module_Reuse (I22), Network_Type (I23). In the neural network model, the qualitative factors are represented as 0 or 1. So qualitative factors do not make a big difference from the quantitative factors in computations.

〈Table 1〉 Definition of Input Factors

Categories	Input Factors	Values	Type
Project Features	Duration Constraints(I1)	[Normal | Urgent]	Qualitative
	Duration of Training and Education(I2)	Average Required Man-Months	Quantitative
	Size(I3)	No. of Module	Quantitative
	Development Type(I4)	[New Development| Maintenance]	Qualitative
Product Features	Processing Type(I5)	[Batch | On-Line]	Qualitative
	Used Algorithms :		
	• D/B Access(I6)	Percentage	Qualitative
	• I/O Process(I7)	Percentage	Qualitative
	• Math./Statistic Process(I8)	Percentage	Qualitative
	• AI based Process(I9)	Percentage	Qualitative
	Task Type :		
	• Strategic(I10)	Percentage	Quantitative
	• Administrative(I11)	Percentage	Quantitative
	• Operational(I12)	Percentage	Quantitative
Staff Features	Skill Level :		
	• Beginner(I13)	Number of Person	Quantitative
	• Junior(I14)	Number of Person	Quantitative
	• Senior(I15)	Number of Person	Quantitative
	Similar Application Development Experience :		
	• Beginner(I16)	Avg. of Months	Quantitative
	• Junior(I17)	Avg. of Months	Quantitative
	• Senior(I18)	Avg. of Months	Quantitative
Technology Features	Development Tool(I19)	[Use | Not Use]	Qualitative
	Methodology(I20)	[Use | Not Use]	Qualitative
	Language(I21)	[3GL | 4GL]	Qualitative
	Module Reuse(I22)	[Use | Not Use]	Qualitative
	Network Type(I23)	[Centralized | Distributed]	Qualitative

3.2 FEEDFORWARD NEURAL NETWORK MODEL

We designed the neural network model and experimented with it to compare with other models, such as the COCOMO model. The feedforward neural network model designed for this study had 23 input factors, one output factor of effort (man-month), and some hidden nodes. The selection of the number of hidden nodes is often based on a heuristic approach (from n/2 to 2n+1, where n means the number of input nodes). The model was trained by using backpropagation algorithm implemented in the system UNIK-NEURO.



(Fig. 1) Full NN Model

We collected data from 45 software projects for training and testing. 35 projects were used for training data, and 10 projects were used for test data. We tested a total of 30 estimates of the randomly selected target projects three times through experiments. Through our experiments, we were able to produce the best design for a neural network model which had one hidden layer, 18 hidden layer nodes and 750 times of learning epochs. The 10 projects were randomly selected for each experiment.

In order to compare it with other estimation model, COCOMO was also designed with the same input variables as the neural network model used. To measure the comparative performance between the two models, we needed to compute the MRE (Magnitude of Relative Error) of each model. This metric was suggested by both Thebaut (1983) and Conte et. al., (1986) and used by Kermerer (1987) for effort model validation. MRE is calculated as the absolute percentage error of the estimate with respect to the actual amount of effort required to complete a project ;

$$MRE = \frac{|MM_{est} - MM_{act}|}{MM_{act}} * 100$$

Where MM_{est} is the estimate of man-months and MM_{act} is the actual effort consumed by the project.

Later, this formula was used to evaluate the accuracy of estimation. A comparison of the MRE between two models is shown in <Table 2>.

〈Table 2〉 Comparison of MRE Between 2 models

MRE \ Models	Revised COCOMO	Naive Neural Network
Average	93%	27%
Standard Deviation	111%	23%
Maximum	540%	86%
Minimum	2.8%	1.5%

A paired t-test between the NN model and COCOMO was performed to validate comparative performance statistically.

- **Revised COCOMO vs NN model**
  - Null Hypothesis (H_0) : $U_{RC} = U_{NN}$
  - Alternative Hypothesis (H_1) : $U_{RC} > U_{NN}$
    (U_{NN} : Mean of MRE by NN model, U_{RC} : Mean of MRE by Revised COCOMO)
  - Paired t-test comparison

Statistics Items	Significance Level	Decision
t-value : 3.16 Degree of Freedom : 58	0.5%	Reject H_0

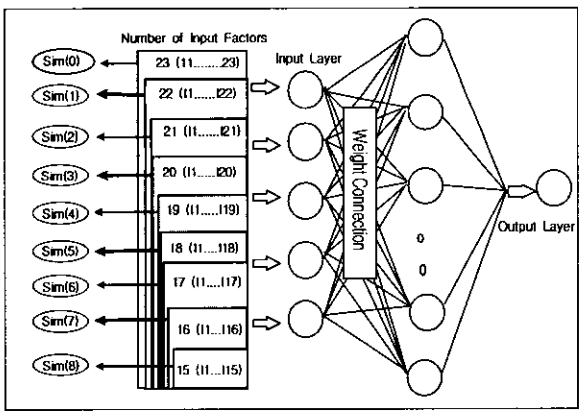
This means that the neural network model is superior to the revised COCOMO.

4. NEURAL NETWORK MODEL AND CASE BASED APPROACH

4.1 PARSIMONIOUS TYPE OF NN MODEL

Qualitative improvements in estimation accuracy can not be achieved by a naive neural network model alone. The most accurate experts utilize a form of similar problem solving called Case-Based Reasoning Approach, in which effort estimation is driven by recall of previously encountered software projects. If we reduce the data noises by matching a target project to past cases, then the NN model is a more efficient approach for increasing the accuracy of estimation. A more important distinction in this study is that the cases with the same qualitative values can be grouped as a case set. Within a case set, we can eliminate the input factors that have the same values making a parsimonious neural network model.

In a parsimonious NN model, input factors used for classifying similar case groups are excluded because those



(Fig. 2) Parsimonious NN Model

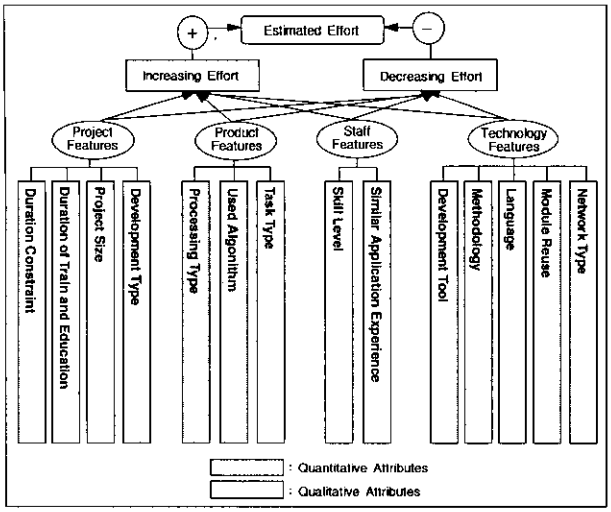
factors have the same values within the case groups. Therefore, the number of input nodes could be reduced from 23 to 15. We can estimate the result with less input factors than those of the naive NN model, so the architecture of the NN model will be changed into a more parsimonious form as in (Fig. 2).

With this type of NN model, the learning time for the final test can be reduced and the accuracy of estimation can be increased because of preprocessing and filtering the noises of input data.

4.2 PARSIMONIOUS NN MODEL COMBINED WITH CBA

To increase the accuracy of estimation of a NN model, the data noises should be eliminated. Several research projects have studied effective estimation performance using an analogy approach of past cases, and most of these studies reported the desired output. A CBR approach was suggested for a software effort estimation model by Mukhopadhyay et al. (1992) by obtaining protocols from a human expert. From a library of cases developed from expert-supplied protocols, an instance called the source is retrieved that is most "similar" to the target problem to be solved. The solution of the most similar problem retrieved from the case library is adapted to account for differences between the source problem and the target problem using rules inferred from analysis of the human expert's protocols. CBR system can be used to perform more tasks than just classification, neural networks and pattern recognition can not. Main et al. (1995) suggested the use of neural networks for case-retrieval in a system for a fashion shoe design. Basically, we designed the neural network model architecture, and combined it with a case based approach, which we call 'hybrid type of estimation model architecture'. We have to consider the various factors affecting the effort for software development and maintenance. The research architecture required that all factors of the four categories be classified into two types : quantitative factors and qualitative factors, as (Fig.3).

The quantitative factors are mainly used for the neural network model and the qualitative factors are used for increasing the accuracy of estimation in the case-based approach. The qualitative attributes for our estimation model are processed by a case based approach because it is difficult to accurately reflect them in the quantitative problem. It is very convenient to use for the criteria of case group classification by them because the qualitative variables have discrete values. The reflection of quantitative factors for estimation are mainly



(Fig. 3) Type of Input Factors

performed by a NN model architecture. But, there are some problems to classify case groups as they have continuous values. Our major research issues in the case-based approach are not just implementing case-based reasoning in order to find the case which is the most appropriate among past cases, but to also find a group of several cases in each level of similarity to the target case and building an efficient neural network parsimoniously and compactly. Also, in this research, the sensitivity analysis between the case based approach and the neural network model for finding the optimal estimation result is accomplished. In order to accomplish this, the first step is to compute the level of similarity between the target project and past projects. The second step consist of grouping similar projects for input factors of a NN model. And finally, we evaluate the test results of each group by changing the similarity level until we find the optimal results.

4.3 SYSTEM ARCHITECTURE AND PROCEDURE

The system architecture for software effort estimation is suggested in (Fig. 4). In this architecture, we construct the hybrid systems which is combined with NN architecture and a case based approach.

In order to estimate the effort required for the target software project, the characteristics of a target project should be defined, and the input factors necessary for searching similar cases should be selected. Then the cases according to the level of similarity can be grouped. Next, the effort is estimated by the NN model with the changing of the level of similarity intensity. Lastly, in order to find the optimal solution, we analyze the sensitivity of the estimated output of the NN model according to the level of similarirty intensity.

The procedures of software effort estimation by the NN model and CBA are as follows :

Step 1) Define the input factors of a target S/W project estimation

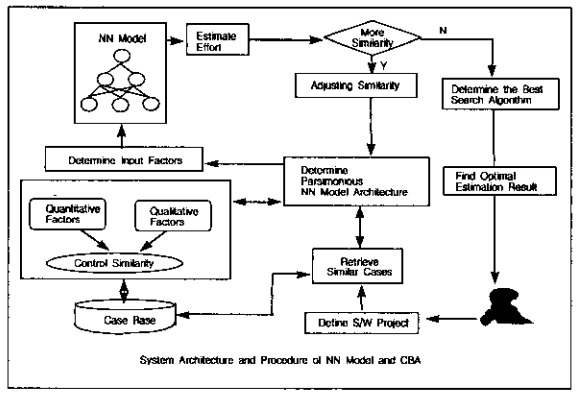
We use the frame representation for defining the target project.

```
{{ New_Process_Mgt
   IS-A : Business_Transaction
   Project_Feature :
     (Development_Type New_development)
     (Duration_Constraints Urgent)
     (Duration_of_Train_Educ (M-M 1))
     (No_of_Modules 240)
   Product_Feature :
     (Processing_type On_line)
     (Used_Algorithm (D/B 1) (I/O 1) (M&S 1) (AI 0) )
     (Task_Type (Strategic 0.5) (Administrative 0.3)
                                (Operational 0.2 ))
   Staff _Feature :
     (Skill_level (S 1) (J 2) (B 4))
     (Avg_Mon._of_Development (S 0) (J 24) (B 16))
  Technology_Feature :
     (Development_Tool Use)
     (Methodology Use)
     (Language 4GL)
     (Network_type Distributed)
     (Module_Reuse Use)
  Actual_Efforts :
     (Man-Month ) }}
```

Step 2) Measure similarity for CBA

We defined the similarity function for the sensitivity analysis, that is how level the case is similar to relatively. The function is defined as follows :

- SIM(Intensity) = Number of the same qualitative input factors



(Fig. 4) System Architecture and Procedure of NN Model and CBA

Here, we define the measure method of similarity of qualitative factors as suggested by Cain (1993) and O'Leary (1993). As in the case of non-numeric values, the nearest neighbor algorithm assumes that a case will be represented

as a set of factors. The similarity metric of this algorithm simply counts the number of factors that a target and a stored case have in common :

$$\text{Intensity} = (\sum_{i=1}^{n} \text{sim}(Case_i, Target_i))$$

where Target_i is a qualitative input factor of the target project and Case_i is that of the stored project cases and sim (x, y) is defined as :

$$\text{sim}(x, y) = 1 \text{ if } x = y$$
$$0 \text{ if } x \neq y$$

Step 3) Set and group the relevant cases according to the feature hierarchy of the target project. For example, in case of setting the similar case group for a target project, Project_domain is Business_Transaction then ;

```
IF    (Development_Type is New_development)
  AND  (Duration_Const is Urgent)
          ⇒ Similarity Intensity of Project Features is 2
IF    (Processing_Type is On-line)
          ⇒ Similarity Intensity of Product Features is 1
IF    (Development_ Tool is Use)
  AND (Methodology is Use)
  AND (Language is 3GL )
  AND (Module_Reuse is Yes)
  AND (Network_Type is Distributed)
          ⇒ Similarity Intensity of Technology Features is 5
    Total SIM intensity is 8 ➔ SIM(8)
```

In the above case, the total similarity intensity becomes 8. The similarity distance is defined as the value of total SIM (8). For example, If the value of SIM is 8, then the relevant cases are retrieved and grouped into a similar case group for an estimation NNmodel.

Step 4) Determine input factors of NN model according to the level of similarity

In the same case group, similar factors, especially qualitative input factors are eliminated from the candidate input factors for NN model because they have the same values within a same case group.

Step 5) Build and train NN models with similar cases

The number of cases in a same case group are used for the instances of NN model to estimate effort

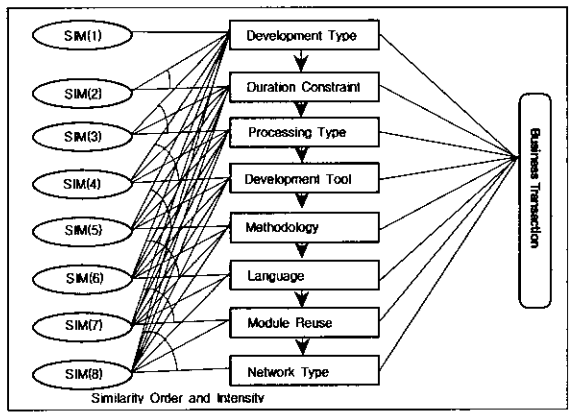
Step 6) If the level of similarity are found no more, then go to Step 8) Else go to Step 7)

Step 7) Increase the intensity level of similarity and go to Step 3)

Step 8) Determine the best search algorithm and find the optimal estimation result

4.4 SIMILARITY PRIORITY AND INTENSITY

In order to define the similar case group, we determined the priority order of qualitative input factors and defined the intensity of similarity based on analyzing the result of survey in the previous section.



(Fig. 5) Similarity Order

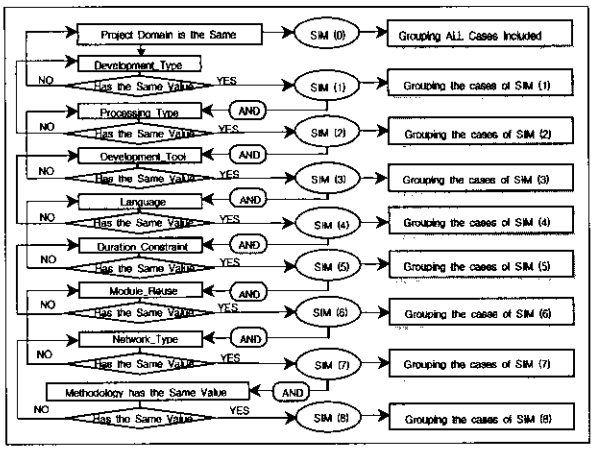
(Fig. 5) shows the hierarchies of major qualitative input factors in software project on business transaction. Here, we define the priority order of similarity factors according to the importance of qualitative input factors in the survey as mentioned before.

For example, if S/W Project for Marketing Management is selected as a target project, we could group the similar cases as comparing the similar input factors according to the priority and classify the similar case groups by each intensity of similarity.

Similarity intensity is computed by SIM function sequentially by the priority order as follows :

1) Project_Domain is Business Transaction : All cases which have SIM (0) that are used
2) Development_Type is New_Development : Case group has SIM (1)
3) Duration_Constraint is Urgent : Case group has SIM (2)
4) Processing_Type is On-Line : Case group has SIM (3)
5) Development_Tool is used : Case group has SIM (4)
6) Methodology is used : Case group has SIM (5)
7) Language is 4GL : Case group has SIM (6)
8) Module_Reuse is used : Case group has SIM (7)
9) Network_Type is Distributed : Case group has SIM (8)

In this paper, the algorithms of grouping the similar cases according to similarity intensity is shown in (Fig. 6).



(Fig. 6) Algorithms of Grouping Similar Cases

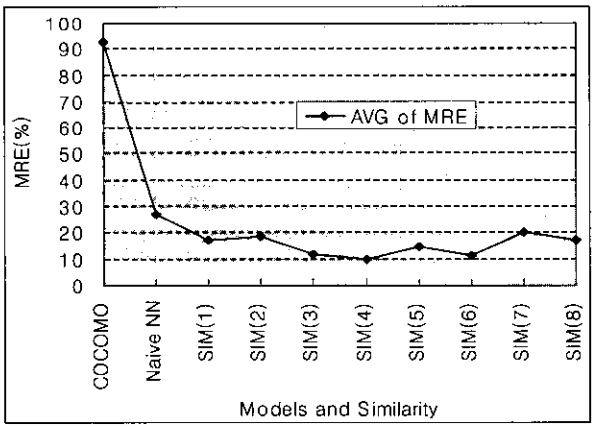
The number of similar cases within the same case group are collected according to the similarity level above mentioned in <Table 3>. Generally, between the number of similar cases and similarity level, there are two relationships : The high similarity intensity with few cases, and the low similarity with more cases. This means that the lower similarity, the more cases it will have ; and on the contrary the higher similarity, the less cases it will have.

〈Table 3〉 The Number of Similar Cases

Similarity	All Cases	SIM (1)	SIM (2)	SIM (3)	SIM (4)	SIM (5)	SIM (6)	SIM (7)	SIM (8)
No. of Cases	45	33	26	20	13	10	10	5	5

4.5 SENSITIVITY ANALYSIS

In the case of estimating the effort of a target project, we need to define the similar factors and search the similar cases. Then we compute the similarity intensity by using a similarity function, and estimate the effort by using the NN model on each case group with the same similarity.



(Fig. 7) Sensitivity Analysis of MRE and Similarity Level

It is possible to analyze the sensitivity for getting the optimal solution which has the least MRE. (Fig. 7) is a graph representing the relationship between the MRE and the level of similarity.

In our test, we found a relationship between the level of similarity intensity and the estimates. As shown in the graph representation of (Fig. 8), the MRE of the estimated value of the target project fluctuates with the level of similarity. This sensitivity analysis continues the estimation process until it finds the final desired result by changing the degree of similarity level and simultaneously adjusting input factors of NN architecture.

5. PERFORMANCE EVALUATION

5.1 COMPARISON OF EXPERIMENT RESULTS

In order to compare our NN/CBA model with a naive neural network model and COCOMO, we tested the output with the same data set used in 3.3. We conducted experiments three times by getting 10 estimates of the randomly selected projects each time. The total 30 results of software project estimation are used for comparison of estimation performance.

Each of the three experiments randomly tested 10 projects using 35 project cases and we got the estimates of 30 total projects as showed in <Table 4>.

〈Table 4〉 Effort Estimates by Each Model

Project	Estimated Effort(MM)										
	Actual Values	COC-OMO	Naïve NN : SIM(0)	NN Combined with Similar Cases (Similarity Intensity)							
				SIM (1)	SIM (2)	SIM (3)	SIM (4)	SIM (5)	SIM (6)	SIM (7)	SIM (8)
Pro. 1	30	62	23	22	24	26	23	25	27	35	26
Pro. 2	15	57	19	16	18	14	13	12	14	17	12
Pro. 3	20	25	36	28	26	25	17	15	16	21	23
Pro. 4	50	63	43	43	42	44	51	42	45	33	33
Pro. 5	62	103	48	45	38	51	55	66	72	42	65
Pro. 6	36	75	34	41	45	33	33	48	42		
Pro. 7	23	18	32	28	21	23	25	22	22		
Pro. 8	10	33	12	8	11	9	11	9	11		
Pro. 9	8	18	14	9	10	11	8	8	9		
Pro.10	22	15	41	28	23	18	19	25	23		
Pro.11	98	263	167	103	79	67	88				
Pro.12	14	23	16	20	21	13	15				
Pro.13	30	27	42	43	26	32	28				
Pro.14	36	35	39	39	33	35					
Pro.15	40	84	42	44	36	48					
Pro.16	48	62	35	50	44	46					
Pro.17	102	73	68	84	86	97					
Pro.18	33	54	32	34	32	32					
Pro.19	67	98	68	59	70	67					
Pro.20	80	102	120	78	88	82					

Pro.21	13	22	15	16	14						
Pro.22	420	640	520	578	563						
Pro.23	12	53	15	13	18						
Pro.24	49	27	53	55	53						
Pro.25	5	32	6	7	6						
Pro.26	24	20	23	23	19						
Pro.27	9	12	11	10							
Pro.28	150	230	136	128							
Pro.29	8	16	7	8							
Pro.30	9	24	11	10							

To measure the comparative performance of our model, MRE values from the outputs of each model need to be computed through the experiment as shown in <Table 5>. In the following table, the dark cells show the best MRE which has the minimum error for each project. We computed the statistics of these as follows :

Statistics of the minimum MRE by each project			
Average : 6.45	Standard Deviation : 5.34	Max : 24	Min : 0

The average of the best MRE value is 6.45%. Therefore, we need a efficient search algorithm to find this value for each project. In 5.2, we devise theorems of search algorithm for finding it. In <Table 5>, the values of MRE in each similarity are showed.

〈Table 5〉 Values of MRE (Unit : %)

Project	COCOMO	Naive NN	SIM (1)	SIM (2)	SIM (3)	SIM (4)	SIM (5)	SIM (6)	SIM (7)	SIM (8)
Pro. 1	107	23	27	20	13	23	17	10	17	13
Pro. 2	67	27	6.7	20	6.7	13	20	6.7	13	20
Pro. 3	25	80	40	30	25	15	25	20	5	15
Pro. 4	26	14	14	16	12	2	16	10	34	34
Pro. 5	66	23	27	39	18	11	6.5	16	32	4.8
Pro. 6	108	5.6	14	25	8.3	8.3	33	17		
Pro. 7	22	39	22	8.7	0	8.7	4.3	4.3		
Pro. 8	230	20	20	10	10	10	10	10		
Pro. 9	125	75	13	25	38	0	0	13		
Pro.10	32	86	27	4.5	18	14	14	4.5		
Pro.11	168	70	5.1	19	32	10				
Pro.12	64	14	43	50	7.1	7.1				
Pro.13	10	40	43	13	6.7	6.7				
Pro.14	2.8	8.3	8.3	8.3	2.8					
Pro.15	110	5	10	10	20					
Pro.16	29	27	4.2	8.3	4.2					
Pro.17	28	33	18	16	4.9					
Pro.18	64	3	3	3	3					
Pro.19	46	1.5	12	4.5	0					
Pro.20	28	50	2.5	10	2.5					
Pro.21	69	15	23	7.7						
Pro.22	52	24	38	34						
Pro.23	342	25	8.3	50						
Pro.24	45	8.2	12	8.2						
Pro.25	540	20	40	20						
Pro.26	17	4.2	4.2	21						
Pro.27	33	22	11							
Pro.28	53	9.3	15							
Pro.29	100	13	0							
Pro.30	167	22	11							

The total trends of MRE are represented as shown in <Table 6>. We analyzed and evaluated the sensitivity between the values of MRE and the intensity of case similarity.

〈Table 6〉 Statistics of MRE Values

Statis-tics	COCO -MO	Naive NN	NN Combined with Similar Cases (Similarity Intensity)							
			SIM (1)	SIM (2)	SIM (3)	SIM (4)	SIM (5)	SIM (6)	SIM (7)	SIM (8)
Avg. of MRE	93%	27%	17%	19%	12%	10%	15%	11%	20%	17%
STD of MRE	111%	23%	13%	13%	10%	5.9%	10%	5.3%	13%	11%
MAX MRE	540%	86%	43%	50%	38%	23%	33%	20%	34%	34%
MIN MRE	2.8%	1.5%	0%	3%	0%	0%	0%	4.3%	5%	4.8%

Through the empirical test, we were able to find the best result, which has the least MRE value. The best estimate is as follows :

Similarity Intensity is SIM (4) which has a MRE average of 10%, Standard deviation is 5.9%, Max MRE is 23% and Min MRE is 0%. Comparing the results reveals that the combined model with NN and CBA reduced the errors of the naive NN model. But, we can not use the case group which has SIM (4) as the learning data for NN model. A target project of SIM (4) may have no similar cases because it missed the estimate of 17 projects (from pro. 14 to pro. 30) as in <Table 4>. So, we need to devise rules to get the estimate in such a case.

5.2 SEARCH RULES FOR SENSITIVITY ANALYSIS

We defined three search rules for sensitivity analysis to get the least value of MRE instead of using the average of MRE by each level of similarity.

5.2.1 Theorem of Search Rule 1

- The similarity level which has the point of the first convex is selected as the best solution for an estimation model.
- When several points which have the same values exist on the convex points,

we selected the most right-hand side value.

5.2.2 Theorem of Search Rule 2

- The similarity level, which has the point of the second convex, is selected as the best solution for estimation model.

- When the several points which have the same values are existing on the convex points, we selected the most right-hand side value.
- If there is no second convex point, we selected the first convex point as in the case of search rule 1.

5.2.3 Theorem of Search Rule 3

- The similarity level, which has the point of the third convex, is selected as the best solution for the estimation model.
- When several points, which have the same values, exist on the convex points, we selected the most right-hand side value.
- If there is no third convex point, we selected the second convex point as in the case of scarch rule 2.
- If there is no second convex point, we selected the first convex point as in the case of search rule 1.

We compared the differences among the best MRE, the MRE of naive NN, and the MREs by Search Rule 1, Search Rule 2 and Search Rule 3 about 30 projects.

In <Table 7>, we can determine search rule 2 as the best rule for searching the least minimum error for getting the optimal output of sensitivity analysis using similar case-based approach.

〈Table 7〉 The Differences between Search Rules and Best MRE

Project	Best MRE (BM)	Naïve NN	Diff.= |BM-NN|	Search Rule1	Diff.= |BM-SR1|	Search Rule2	Diff.= |BM-SR2|	Scarch Rule 3	Diff.= |BM-SR3|
Pro. 1	10	23	13	23	13	10	0	13	3
Pro. 2	6.7	27	20.3	6.7	0	6.7	0	6.7	0
Pro. 3	5	80	75	15	10	5	0	5	0
Pro. 4	2	14	12	14	12	2	0	10	8
Pro. 5	4.8	23	18.2	23	18.2	6.5	1.7	4.8	0
Pro. 6	5.6	5.6	0	5.6	0	8.3	2.7	8.3	2.7
Pro. 7	0	39	39	0	0	4.3	4.3	4.3	4.3
Pro. 8	10	20	10	10	0	10	0	10	0
Pro. 9	0	75	75	13	13	0	0	0	0
Pro.10	4.5	86	81.5	4.5	0	4.5	0	4.5	0
Pro.11	5.1	70	64.9	5.1	0	10	4.9	10	4.9
Pro.12	7.1	14	6.9	14	6.9	7.1	0	7.1	0
Pro.13	6.7	40	33.3	6.7	0	6.7	0	6.7	0
Pro.14	2.8	8.3	5.5	2.8	0	2.8	0	2.8	0
Pro.15	5	5	0	5	0	5	0	5	0
Pro.16	4.2	27	22.8	4.2	0	4.2	0	4.2	0
Pro.17	4.9	33	28.1	4.9	0	4.9	0	4.9	0
Pro.18	3	3	0	3	0	3	0	3	0
Pro.19	0	1.5	1.5	1.5	1.5	0	0	0	0
Pro.20	2.5	50	47.5	2.5	0	2.5	0	2.5	0
Pro.21	7.7	15	7.3	15	7.3	7.7	0	7.7	0
Pro.22	24	24	0	24	0	34	10	34	10
Pro.23	8.3	25	16.7	8.3	0	8.3	0	8.3	0
Pro.24	8.2	8.2	0	8.2	0	8.2	0	8.2	0
Pro.25	20	20	0	20	0	20	0	20	0
Pro.26	4.2	4.2	0	4.2	0	4.2	0	4.2	0
Pro.27	11	22	11	11	0	11	0	11	0
Pro.28	9.3	9.3	0	9.3	0	9.3	0	9.3	0
Pro.29	0	13	13	0	0	0	0	0	0
Pro.30	11	22	11	11	0	11	0	11	0
TOTAL Difference			613.5		81.9		26.6		32.9
AVG of Difference			20.45		2.73		0.88		1.09
No. of 0's Difference			8		22		24		24

According to the above theorems for search rules of the sensitivity analysis, the number of 0's was counted and the percentage was computed by each similarity level in <Table 8>.

The case group that has the SIM (3) of similarity intensity is showed as the best input learning cases for the neural network model.

〈Table 8〉 The Number of 0's in Each Similarity Level

	Naïve NN	SIM (1)	SIM (2)	SIM (3)	SIM (4)	SIM (5)	SIM (6)	SIM (7)	SIM (8)
No.of 0's differences	8	10	6	11	5	2	3	1	1
Total Cases	30	30	26	20	13	10	10	5	5
Percentage	27%	33%	23%	55%	38.4%	20%	30%	20%	20%

5.3 HEURISTIC SEARCH RULES FOR OPTIMAL ESTIMATES OF TARGET PROJECT

It is impossible to compute the MRE of each future target project, so we need to devise another search method that is similar to the previous search algorithms. We extended the three previous theorems to six heuristic search rules for search algorithms to find the least variance between the actual values and the estimated values in each level of similarity intensity.

These theorems assume that the best result exists in convex points or concave points in each level of similarity intensity (i.e., from naive neural network, which means SIM (0) to the highest level of similarity intensity, which means SIM (8)) of each target project.

- Search Rule 1 (1st Convex Point) : select the first convex point as the optimal estimate and the best case group for machine learning
- Search Rule 2 (2nd Convex Point) : select the second convex point as the optimal estimate and the best case group for machine learning
- Search Rule 3 (3rd Convex Point) : select the third convex point as the optimal estimate and the best case group for machine learning

- Search Rule 4 (1st Concave Point) : select the first concave point as the optimal estimate and the best case group for machine learning
- Search Rule 5 (2nd Concave Point) : select the second concave point as the optimal estimate and the best case group for machine learning
- Search Rule 6 (3rd Concave Point) : select the third concave point as the optimal estimate and the best case group for machine learning

Using the previous cases, the search rule was selected as the best search algorithm to get an optimal estimate under the real situations. <Table 9> shows the results got by using these six search rules of search algorithm for finding the best estimate.

<Table 9> Optimal Estimates in Each Search Rule (Unit : MM)

Project	Act. Value	Best Estimate	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
Pro. 1	30	27	22	23	26	23	26	35
Pro. 2	15	14	16	12	12	19	18	17
Pro. 3	20	23	15	15	15	36	23	23
Pro. 4	50	51	42	42	33	43	51	45
Pro. 5	62	65	38	42	42	48	72	65
Pro. 6	36	34	34	33	42	45	48	48
Pro. 7	23	23	21	22	22	32	25	25
Pro. 8	10	(9),(11)	8	9	9	12	11	11
Pro. 9	8	8	9	8	8	14	11	9
Pro. 10	22	23	18	23	23	41	25	25
Pro. 11	98	103	67	67	67	167	88	88
Pro. 12	14	13	16	13	13	21	15	15
Pro. 13	30	(32),(28)	42	26	28	43	32	32
Pro. 14	36	35	33	33	33	39	35	35
Pro. 15	40	42	42	36	36	44	48	48
Pro. 16	48	(50),(46)	35	44	44	50	46	46
Pro. 17	102	97	68	68	68	97	97	97
Pro. 18	33	(32),(34)	32	32	32	34	34	34
Pro. 19	67	67	59	67	67	68	70	70
Pro. 20	80	(78),(82)	78	82	82	120	88	88
Pro. 21	13	14	15	14	14	16	16	16
Pro. 22	420	520	520	563	563	578	578	578
Pro. 23	12	13	13	13	13	15	18	18
Pro. 24	49	53	53	53	53	55	55	55
Pro. 25	5	6	6	6	6	7	7	7
Pro. 26	24	23	19	19	19	23	23	23
Pro. 27	9	10	10	10	10	12	12	12
Pro. 28	150	136	128	128	128	136	136	136
Pro. 29	8	8	7	7	7	8	8	8
Pro. 30	9	10	8	10	10	11	11	10
No. of the least minimum values			8	13	14	6	11	12

As showing in <Table 9>, dark shaded cells represent the least errors of man-months between the actual MM and the estimated MM. Therefore, we found Search Rule 3 as the best rule for search algorithm in the empirical test. We computed the statistics of MRE for the 30 total target projects, which were randomly selected as shown in the following table.

Average of MRE	14.3%
Standard Deviation of MRE	11%
Max MRE	34.7%
Min MRE	0%

It means that this result is more efficient than not using search algorithm, that is to say, the total average of MRE (15.2%) that is computed by each level of similarity intensity as showed in <Table 10>. It shows the comparative statistics of MRE by using the existing estimation model (COCOMO), a naive neural network model, a combined model of neural network and case based approach, and a method of the suggested heuristic search algorithm approach.

<Table 10> Comparative Comparison of MRE in Each Model and Method

	COCO-MO	Naïve Neural Network	Average of Each Similarity Level	Search Rule 3 of Search Algorithm	Minimum Values of Each Similarity Level
Avg of MRE	93%	27%	15.2%	14.3%	6.5%
STD of MRE	111%	23%	10.2%	11%	5.3%
Max MRE	540%	86%	50%	34.7%	24%
Min MRE	2.8%	1.5%	0%	0%	0%

It means that the estimation model of neural networks with case based approaches has a more efficient estimation performance by using the heuristic rule for search algorithm as in <Table 10>. Therefore, we could summarize that we have to use the neural network model combined with case based approaches supported by rules for heuristic search algorithm in order to achieve a more accurate estimate.

6. CONCLUSION

To overcome the drawbacks of the existing software effort estimation models for the rapidly changing software development or maintenance environment, we suggested a new estimation model which we called the hybrid effort estimation model as it is composed of the neural network model and the case-based approach supported by the heuristic search algorithm. It is possible to test the sensitivity of the estimated

values of the neural network model by changing the number of the input factors and the similarity intensity. With this mechanism, the neural network model can be designed and built more parsimoniously and efficiently. An efficient heuristic search rule needs to be devised for finding the best estimate of our estimation model automatically and for classifying the types of relationship in the estimated value according to the level of similarity intensity. The preliminary results obtained from our experiment show that the neural network model performed effectively with the assistance of a case based approach using optimal similar cases from the past projects. We also described the estimation procedures and suggested the efficient search algorithms for finding the best output of effort for a target project. In future studies, we are going to verify that the model is a promising, and versatile tool for accurately estimating software effort with more collected cases.

REFERENCES

[1] Abdel-Hamid, T. K., "Adapting, Correcting and Perfecting Software Estimates : A Maintenance Metaphor," IEEESoftware, Vol.5, No.4, pp.15-22, Jul. 1988.

[2] Barletta, B., "An Introduction to Case-Based Reasoning," AI Expert, pp.43-49, Mar. 1992.

[3] Bergeron, F., and St-Arnaud, J. Y., "Estimation of Information Systems Development Efforts," Information and Management Vol.22, pp.239-254, 1992.

[4] Boehm, B. W., "Software Engineering Economics," Tutorial : Software Management (3rd ed.), IEEE Computer Society, pp.135-152, 1984.

[5] Boehm, B. W., "Understanding and Controlling Software Costs," IEEE Trans. on Soft. Eng., Vol.14, No.10, pp. 1462-1477, Oct. 1988.

[6] Hill, T., O'Connor, M. L. and Remus, M., "Artificial Neural Network Models for Forecasting and Decision Making," International Journal of Forecasting, pp.5-15, Sep. 1994.

[7] Jun, E. S, "Using Artificial Neural Network for Software Development Efforts Estimation," KISS Vol.3, No.1, pp. 211-224, Jan. 1996.

[8] Kermerer, C. F., "An Empirical Validation of Software Cost Estimation Models," Commun. Of the ACM, Vol.30, No.5, pp.416-429, May. 1987.

[9] Kermerer, C. F., "An Empirical Validation of Software Cost Estimation Models," Commun. Of the ACM, Vol.30, No.5, pp.416-429, May. 1987.

[10] Lee, Hoeseok, "A Structured Methodology for Software Development Effort Prediction Using the Analytic Hierarchy Process," Journal of Systems Software, Vol.21, pp.179-186, 1993.

[11] Lee, Jae Kyu and Kim, Ho Dong, "Man-hours Requirement Estimation for Assemblies using Neural Networks," 94 Japan/Korea Joint Conference on Expert Systems, Tokyo, Mar. pp.22-24, 1994.

[12] Main, J., Dillon, T. S. and Khosla, R., "Use of Neural Network for Case-Retrieval in a System for Fashion Shoe Design," Industrial and Engineering Applications of AI and Expert Systems, Proceedings of the 8th International Conference, pp.151-158, 1995.

[13] Mukhopadhyay, T., Vicinanza, S. S., and Prietula, M. J., "Examming the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation," MISQ (4 : 1), pp.155-171, Jun. 1992.

[14] Saiedian, H., Zand, M., and Barney, D., "The Strengths and Limitations of Algorithmic Approaches to Estimating and Managing Software Costs," International Business Schools Computing Quarterly, Spring, pp.21-22, 1992.

[15] Srinivasan, K. and Fisher, D., "Machine Learning Approaches to Estimation Software Development Effort," IEEE Trans. On Soft. Eng., Vol.21, No.2, pp.126-137, Feb. 1995.

[16] Venkatachalam, A. R., "Software Cost Estimation Using Artificial Neural Networks," in Proceedings of 1993 International Joint Conference on Neural Networks, pp. 987-990, Jul. 1993.

[17] Vicinanza, S. S., Mukhopadhyay, T., and Prietula, M. J., "Software-Effort Estimation : An Explolatory Study of Expert Performance," Information Systems Research, Vol.2, No.4, pp.243-262, Dec. 1991.

[18] Wrigley, C. D., and Dexter, A. S., "A Model for Measuring Information System Size," MISQ, Vol.15, No.2, pp.245-257, Jun.1991.

전 응 섭

e-mail : esjun@hanmail.net

2000년 한국과학기술원 정보공학 박사

1985년~1989년 한국과학기술연구소 시스템공학센터 연구원

1989년~1991년 한국HP 시스템컨설턴트

1991년~현재 인덕대학 소프트웨어개발과 교수

관심분야 : 전문가시스템, 인공신경망, 소프트웨어공학, 인터넷 및 전자상거래