Bayesian Belief Networks as a Software Productivity Estimation Tool

S. Bibi, I. Stamelos, L. Angelis

Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece
E-mails: {sbibi,stamelos,lef}@csd.auth.gr

Abstract. Defining the required productivity in order to complete successfully and within time and budget constraints a software development project is actually a reasoning problem that should be modelled under uncertainty. The contribution of this paper is the analysis of the applicability of probabilistic reasoning approaches, in particular Bayesian Belief Networks (BBN), to this problem. BBNs are capable of discovering the dependencies and independencies among the attributes of a project and defining the direct impact of some of them on productivity. Uncertainty is depicted through the use of estimate intervals and probabilities: the estimation is actually an interval within which the productivity of a project is likely to fall in, with a certain probability, considering both an optimistic and a pessimistic situation. The use of predefined intervals is another important feature of the method, allowing the control of the estimation process and the generation of meaningful intervals, appealing and understood by software managers. The ability of the method to classify correctly the rest of the attributes in one of their discrete values is also tested, paying further attention on the software development mode. The method is applied and evaluated on the widely known COCOMO81 dataset. The evaluation shows that BBN is a promising method whose results can be confirmed intuitively. BBN are easily interpreted, allow flexibility in the estimation, can support expert judgment and create models considering all the information that lay in a dataset by including all productivity factors in the final model.

Keywords: Cost estimation, Bayesian Belief Networks, Predefined Intervals, COCOMO.

1 Introduction

Estimating the Productivity of a Software development project remains a challenge for the researchers. Despite the various methods proposed, unsolved questions and problems justify further research and experimentation. The diversity of cost factors, their unclear contribution to productivity and the lack of information in the early stages of software development are the main components of the problem, classifying it to probabilistic reasoning.

The most critical issue in this scientific endeavour is the agreement on the constituent, pertinent elements of the problem. Classical methods demand simple linear structures and a wealth of data often missing in software engineering. A flexible and competitive method to the above methods is Bayesian Belief Networks (BBN). Graphical models such as BBN have become attractive tools because of their ability to efficiently perform reasoning tasks and to represent uncertainty in expert systems [13]. The knowledge that they manage is in the form of dependence and independence relationships, two basic notions in human reasoning.

BBNs deal with uncertainty in various ways. They manage to articulate expert beliefs about the dependencies between different variables and to propagate consistently the impact of evidence on the probabilities of uncertain outcomes. They suggest a structure between the variables of a problem giving the possibility according to the knowledge acquired to determine the starting point of the estimation. Although BBNs can handle continuous variables this is rarely suggested due to practical problems. As a consequence productivity values have to be quantified into categories necessitating the estimation of productivity intervals. Intervals give a pessimistic estimate (maximum value) and an optimistic estimate (minimum value) between which the actual productivity of a project may fall in. With this approach, both uncertainty and risk are considered, weighing the chance of events occurring and the impact they might have. The intervals of productivity have to be pre-defined before the estimate generation. This helps to control the estimation procedure, distribute the projects in the training dataset as uniformly as possible into the various productivity intervals and ensure that estimate intervals will not be too large.

Purpose of this study is to extract useful patterns from cost estimation data with the help of BBNs and to provide some evidence of the prediction accuracy of this technique. Also some conclusions will be drawn concerning the factors that tend to affect productivity directly. An additional research target is the estimation of software development mode based on the values of the rest of the projects attributes. The method is applied on the widely known COCOMO81 dataset.

Studies regarding the use of BBNs in Software Engineering concern mostly Software Quality [11], [20]. In Software Productivity Estimation one study is found [16] where an empirical BBN, based on Boehm's informal classification of COCOMO cost factors was described. In general, various studies have been found suggesting parametric models for software cost estimation, and for evaluating and comparing various methods [1], [5], [8], [10], [15]. In particular some of them suggest the estimation of intervals [7], [9], [17] and one the estimation of predefined intervals [16]. Only one study [19] is found in the literature concerning the estimation of Software development mode.

This paper starts with the description of the dataset in Section 2. Section 3 presents briefly the estimation technique. Section 4 provides the results of BBN in the COCOMO81 dataset. In Section 5, the advantages and disadvantages of BBN as well as future work are discussed.

2 Description of the dataset

The dataset that was used in this analysis is the COCOMO81 [2] dataset coming from TRW defense systems. This dataset consists of 63 projects. A set of 17 attributes, called cost drivers, that is considered to contribute to productivity, and as a consequence to cost, is used in order to extract useful patterns. These attributes, based also on the analysis of Kitchenham in [8], can take the following values: Super Low (SL), Extra Low (EL), Very Low (VL), Low (L), Nominal (N), High (H),Very High (VH), Extra high (EH), Super High (SH). The attributes used are shown in tables 1 to 4:

Table 1. Product attributes

RELY: Required software reliability.	EL, V, L, N, H, VH
DATA: Databases size.	L, N, H, VH, EH
CPLX: Product complexity	VL, L, N, H, VH, EH, SH

Table 2. Computer attributes

TIME: Constraints in the execution time.	N, H, VH, EH, SH
STOR: Constraints in main memory.	N, H, VH, EH, SH
VIRT: Availability of virtual machine.	L, N, H, VH
TURN: Service cycle duration.	VH, H, N, L, VL

Table 3. Personnel attributes

ACAP: Analysts capabilities.	SH, VH, H, N, L, VL, SL
AEXP: Analysts experience.	VH, H, N, L, VL
PCAP: Programmers capabilities.	SH, VH, H, N, L, VL, SL
VEXP: Virtual machine experience.	H, N, L, VL
LEXP: Experience with the language	H, N, L, VL

Table 4. Project attributes

CONT: Personnel’s continuity.	L, N, H
RVOL: Requirement’s volatility.	L, N, H, VH, EH, SH
TOOL: Use of programming tools.	EH, VH, H, N, L, VL
MODP: Use of modern programming practices	EH, VH, H, N, L, VL
SCED: Time schedule	LAX,N, COM,VCOM

Apart from these attributes, the programming language used (LANG), the platform on which the project was developed (PLATFORM), the application type (A.T.) and the software development mode (MODE) are also taken into consideration. These attributes are shown in Table 5:

Table 5. Descriptive attributes

LANG	COBOL, FORTRAN, ADA etc
A.T.	SCI, HMI, SYS, SUP, BUS
PLATFORM	MIN, MAX, MID, MIC
MODE	EMBEDDED, SEMIDETACHED, ORGANIC

3 Bayesian Belief Networks

3.1 Modeling Technique

Bayesian Belief Networks are Directed Acyclic Graphs (DAGs), which are causal networks that consist of a set of nodes and a set of directed links between them, in a way that they do not form a cycle [12]. Each node represents a random variable that can take discrete or continuous finite, mutually exclusive values according to a probability distribution, which can be different for each node. Each link expresses probabilistic cause-effect relations among the linked variables and is depicted by an arc starting from the influencing variable (parent node) and terminating on the influenced variable (child node). The presence of links in the graph may represent the existence of direct dependency relationships between the linked variables (that some times may be interpreted as causal influence or temporal precedence). The absence of some links means the existence of certain conditional independency relationships between the variables.

The strength of the dependencies is measured by means of numerical parameters such as conditional probabilities. Formally, the relation between the two nodes is based on Baye's Rule[6]:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \quad . \tag{1}$$

For each node A with parents B1, B2,…, Bn there is attached an NxM Node Probability Table (NPT), where N is the number of node states and M is the product of its cause-nodes states. In this table, each column represents a conditional probability distribution and its values sum up to 1.

3.2 Methodology

Building a BBN for a domain of application involves three tasks. The first of these is to identify the variables that are of importance, along with their possible values. The second task is to identify the relationships between the variables involved and to express them in a graphical structure. The last task in building a probabilistic network is to obtain the probabilities that are required for its quantitative part. For software productivity estimation all of the above tasks were performed in order to build a BBN.

The data collection was already done by B.Boehm [2]. In COCOMO81 all 63 projects were considered. Data cleaning was not necessary due to the well organized and carefully selected data. All the variables regarding the project, personnel, computer, and project attributes plus the software development mode, the language used, the application type of the project and the platform where the project was developed, participated in the calculations and their value can be assessed (up to some extend) before the completion of the project.

The variables considered up to now are discrete, categorical, and therefore suitable for fitting into a BBN. A problem though arises in productivity that is a continuous variable, which needs to be quantified into discrete valued categories. The choice of the number of intervals and the width of each interval had to be defined. The number of intervals may be selected automatically according to various rules that are proposed in the statistical literature, depending on the number of projects in the dataset and the variance of the actual productivity values, keeping in mind that the narrower the interval, the more useful the estimate is. Sturge's rule [18] was taken into consideration for the determination of the number of the prediction intervals:

$$K = 1 + 3.3 \log n \tag{2}$$

where n is the number of records in the dataset (63). In this case K ≈ 6.9 ≈ 7. We wanted unequal intervals due to the log-normal distribution of productivity values. As a consequence, equal intervals of the logarithm of productivity were taken, classifying each project into one of these categories. The borders of the categories coming from the transformation of the logarithmic values to normal were not rounded but in order to make the intervals chosen appealing and easily identifiable by a human we rounded lower and upper limits without disturbing the distribution of projects into the statistically derived categories. This method has an additional benefit: intervals grow progressively larger, so as to allow larger magnitude errors for higher productivity values. For the COCOMO81 dataset, productivity was counted in delivered source code per man month namely DSI, representing the size of the project, for which 7 intervals were also devised. The productivity intervals are shown in (Table 6).

Table 6. Predefined productivity intervals for the COCOMO81 dataset

Interval number	1	2	3	4	5	6	7
Ln (productivity)	≤ 3.6	3.6-4.18	4.18-4.76	4.76-5.34	5.34-5.92	5.92-6.5	>6.5
Productivity (DSI/MM)	≤ 30	31-65	66-110	111-200	201-375	376-650	>650
Number of projects	3	7	10	14	15	8	6

The process continues with the automated definition of the structure of the network from the training data. For this purpose an heuristic algorithm was applied to the data [4].

The main problem was that this algorithm, apart from the values of the NPT tables, also defines automatically the structure of the BBN. Although this function is very useful for datasets with few fields, few categories for each variable and strong relationships among them, it tends to complicate things when the database includes several fields and many variable categories. In these cases, a very complex Bayesian Network arose where variables, which seemed so far independent, according to the algorithm were found dependent. The only directions that we can give to the algorithm are the order of the nodes, according to which the search of the parents for each node is done. The formula that is used for the construction of the BBN is the following, where D is a database of cases, Z is the set of variables represented by D and Bs_i and Bs_j are two Bayes network structures containing exactly those variables that are in Z:

$$\frac{P(B_{S_i}\mid D)}{P(B_{Sj}\mid D)}=\frac{\dfrac{P(B_{S_i},D)}{P(D)}}{\dfrac{P(B_{Sj},D)}{P(D)}}=\frac{P(B_{Si},D)}{P(B_{Sj},D)} \tag{3}$$

$$P(B_s,D)=P(B_s)\prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}N_{ij}\prod_{k=1}^{r_i}N_{ijk}! \tag{4}$$

$$N_{ij}=\sum_{k=1}^{r_i}N_{ijk} \tag{5}$$

A variable x_i in Z has r_i possible value assignment (v_{ij},......, v_{ir_i}), each variable x_i in B_s has a number of parents who are represented by a list of variables π_i ,q_i represents the unique instantiations of π_i , w_{ij} denotes the jth unique instantiation of π_i, N_{ijk} is the number of cases in D in which variable x_i has the value of v_{ik} and π_i is instantiated as w_{ij}. Further details of k2 algorithm can be found in [4].

Although the suggested BBN from this algorithm indicated interesting relationships among variables, it needed further refinement and processing. Using our intuition, some nodes were inserted and others were deleted especially in the case of productivity node.

4 Results

Before discussing the results, it is useful to define the accuracy metrics that will be used in order to compare and evaluate the results of the model. In particular, Mean Magnitude Relative Error will be used, defined as:

$$MMRE = \frac{100}{n}\sum_{i=1}^{n}\frac{|P_i - E_i|}{P_i} \qquad \textbf{(6)}$$

where P_i is the actual productivity, E_i is the estimate and n is the number of projects. In the case of interval estimates, relative errors are calculated by considering the mean of the interval.

Also, $PRED(Y)$ will be used, i.e. the percentage of projects for which the prediction falls within the Y% of the actual value. In this study, we employ $PRED(25)$.

For the evaluation of the models, *hitrate* will also be used [7], i.e. the percentage of projects for which the correct interval has been successfully estimated. Usually the validation of a model is done by removing one data point at a time from the dataset, recalculating the model and estimating the value of the project that was left out (this method is known as JackKnifing). Also in order to demonstrate the fitting of the model to the data, the same accuracy metrics are presented for the whole dataset (same training and validation data).

In the literature, desired values for MMRE and PRED(25) are 0.25 and 0.75 respectively. However, such values are rarely produced by cost estimation studies because of the difficulty of the problem and the lack of suitable data. We consider that a reasonable desired value for hitrate may be 0.75.

Initially, K2 algorithm [4] was applied to the data in order to have a suggestion of the structure of the model based totally on automated statistical analysis.

The BBN that was extracted is presented in Figure 1 with the help of an open source tool [14].

Each node is accompanied by a number representing fitting hitrate. Algorithms searching for the structure of the model tend to reduce as possible the number of the parents of each node. Typically, up to 5 parents are accepted for each node. It is obvious that in most cases one parent is selected for each node. So in cases that the parent has approximately as many discrete values as the child node or more, hitrates are improved, estimating correctly the 3/5 of the projects according to the predicted attribute (VEXP, PCAP, TIME, CONT). In cases where two parents are considered (SCED, MODP, CPLX, MODE) hitrates are further improved, showing that when more parents are considered more information is exploited for estimating the values of a particular node and as a result the estimation is more successful.

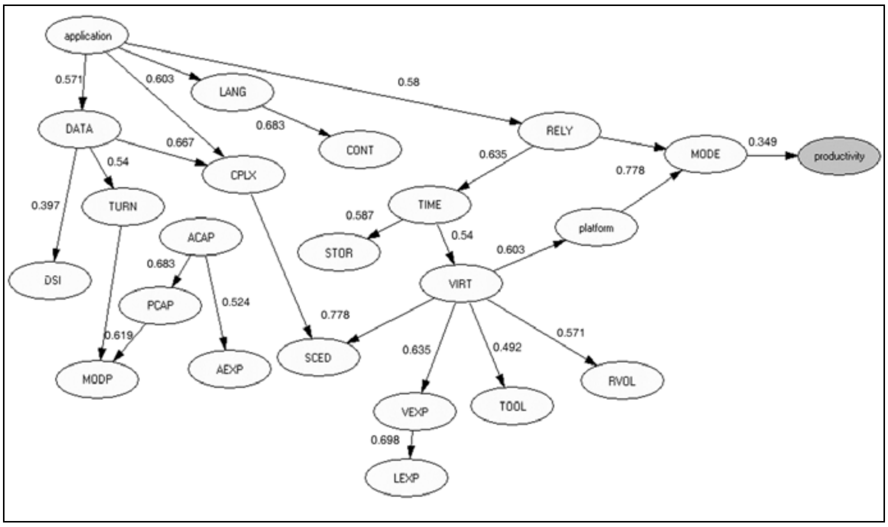


Fig. 1. BBN derived from the original data

In the case of the nodes representing DSI and productivity the results are not so impressive. Regarding productivity this result should be expected, considering the fact that productivity takes 7 discrete values unlike its parent (MODE), that takes only 3 discrete values. According to the conditional probabilities for each discrete value of MODE, one productivity interval is more likely to be noticed in each case, so only three productivity categories are considered and projects belonging to other intervals are misclassified. The same explanation stands up for DSI.

Especially for productivity, it is evident that the parent nodes should be enriched considering some additional nodes. The suggested parent node (MODE) remains. Application type is inserted as a parent node because from the structure it seems that it affects many relevant nodes affecting indirectly productivity. Also CPLX and PCAP are added as parent nodes as they are two of the most representative cost factors of the product and the programmers' attributes correspondingly. Various experimentations on the parent nodes indicated that this particular combination of parents was the most appropriate for distributing the projects into the right productivity interval, as projects with similar values in these attributes in many cases belong to the same productivity interval. The proposed alternative structure having productivity as an external node is presented in Figure 2.

The change in the parent nodes of productivity does not affect the rest of the model due to the conditional independencies. In this model, more information is taken into account when estimating productivity and, as a result, the fitting of the model to the data, as well as its estimation accuracy are improved (Table 7). It should be mentioned that in many cases a particular combination of parents were included only one time in the dataset so when Jackknifing the model was incapable of estimating many projects. However, for the rest 17 cases that can be estimated, it seems that the model classified correctly the majority of the projects.

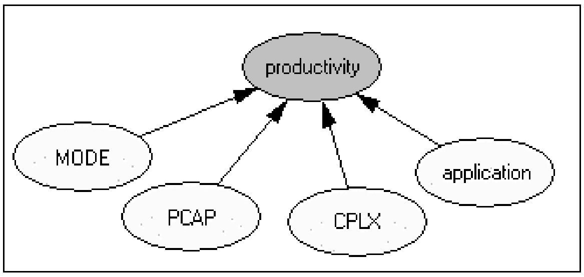


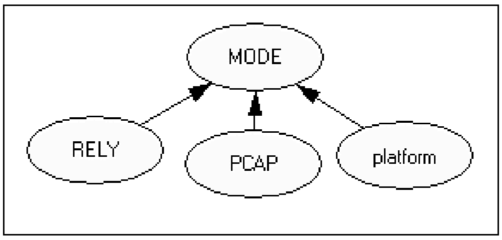
Fig. 2. Productivity node and its parents

Table 7. Accuracy metrics regarding productivity

Productivity	Old model	New model	
	Fitting	Fitting	Accuracy
Hitrate	0.34	0.92	0.7
Pred(25)	0.33	0.81	0.7
MMRE	0.98	0.19	0.32

In the original COCOMO model, Mode is pointed out as an important factor that affects productivity, so it is crucial to find an automated way to assess its value according to the rest of the project attributes. The fitting of the model is satisfying but further improvement can be observed when adding the links shown in Figure 3.

Fig. 3. MODE node and its parents



In (Table 8) the fitting and the accuracy of the initial model and the processed one are presented.

The results in both cases are competitive to the desired values mentioned above and indicate the ability of the BBN to classify correctly the majority of projects to a Mode category. For the estimation of Mode, the model was able to provide an estimation for the majority of projects, 41.

Table 8. Accuracy metrics regarding MODE

MODE	Old model		New model	
	Fitting	Accuracy	Fitting	Accuracy
Hitrate	0.77	0.75	0.91	0.81

In order to cope with the problem of the non-estimated projects, we decided to merge various discrete values of several attributes according to their frequencies and their vicinity. The merges that took place are shown in table 9 and left each attribute with three or four categories.

Table 9. Merges of attribute values on COCOMO81 dataset

Cat.	(1)	(2)	(3)	(4)
RELY	EL, VL	L, N	H, VH	
DATA	L	N,H	VH, EH	
CPLX	VL, L, N	H, VH	EH, SH	
TIME	N	H, VH	EH, SH	
STOR	N	H, VH	EH, SH	
VIRT	L	N	H, VH	
TURN	VL, L	N	H, VH	
ACAP	SL, VL, L, N	H	VH, SH	
AEXP	VL, L	N	H	VH
PCAP	SL, VL, L	N	H, VH	SH
VEXP	VL, L	N	H	
LEXP	VL, L	N	H	
RVOL	L	N	H,VH ,EH,SH	
MODP	VL, L	N,H	VH, EH	
TOOL	VL, L	N	H,VH, EH	

For this case the structure of the BBN is slightly different, many nodes and their parents remain the same, but also some changes and additional links are included, as can be seen in Figure 4. It is obvious that hitrates are improved in most cases, showing that neighbour categories tend to have the same behaviour. It should be mentioned that also in the case of the merged data, when JackKnifing, the model still may not always provide an estimation. This time though 27 cases can be estimated.

No further processing of Mode was considered necessary because the results and the efficiency of the method in the original data was satisfying, covering a great range of the parents' value combinations.

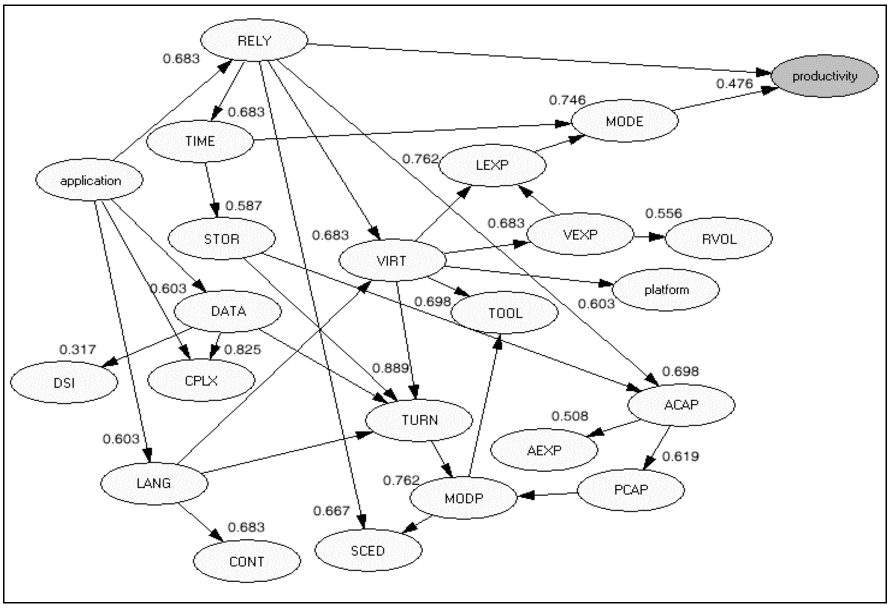


Fig. 4. BBN extracted from the merged data

Considering the same parent set as nodes as in the previous situation for the new model the accuracy metrics are the following (Table 10):

Table 10. Accuracy metrics for the merged data regarding productivity.

Productivity	Old model	New model	
	Fitting	Fitting	Accuracy
Hitrate	0.47	0.86	0.55
Pred(25)	0.42	0.76	0.55
MMRE	0.96	0.26	0.44

5 Conclusions

In this paper various Bayesian Belief Networks have been implemented and evaluated on COCOMO81 dataset. The accuracy of this method and its fitting to the data have been tested. Although no general conclusions can be drawn, some tentative remarks can be made on the suitability of the method, its advantages and drawbacks.

The results coming from the application of BBN to the original COCOMO81 dataset, combining automated statistical methods with expert judgement, are encouraging regarding the suitability of the method in the area of software cost estimation. Even with few attributes, it seems that the combination of the values of some attributes (CPLX, PCAP, MODE and application type) are capable of classifying correctly a project to a productivity interval. Although BBN is a method estimating intervals and should be evaluated exclusively with hitrate, it seems that even when they are forced to give a point estimate, taking the mean of each interval, their results can be considered satisfactory. Also when observing the internal nodes, that represent the attributes that characterize a project and the dependencies among them it seems that most of the links can be confirmed intuitively. In most links among nodes, fitting hitrates are over 60% and further improvement can be expected with the insertion of some additional links. Especially in the case of MODE the results of the fitting of the model, as well as the accuracy metrics are impressive and the suggested structure can have practical use.

BBNs offer a convenient way to solve problems that are not explained logically but rather probabilistically [3]. Software cost estimation is one of these problems: we are not sure of the factors that affect productivity directly and we expect a support from statistical methods to point out the underlying relationships that appear in cost data.

Regarding the advantages of BBN we should mention their ability to combine expert knowledge with past historical, empirical cost data. Expert judgment becomes vital when partial or subjective information is provided about some of the important variables. Also expert knowledge can be used partially for the definition of the models structure, suggesting some factors that directly affect productivity.

Also, it is important that BBNs express uncertainty in many ways. Firstly, by considering productivity intervals, allowing flexibility in the prediction and providing a best and a worst case scenario for productivity during the implementation of the project. Secondly, by assigning to each node a probability distribution given its parents values configuration allowing us to see the strength of each particular combination of values in the training dataset.

BBNs can also integrate partial knowledge and data concerning a project in the form of observed values of some nodes. As a consequence, they can be used as backward reasoning tools in post-mortem analysis, as well as for *what if* analysis in order to explore the impact of changes among productivity factors and to productivity.

In general, BBNs combine visual representation with a strong mathematical background (Bayes theory, Pearl's polytree algorithm, Jensen's junction trees). They are easily interpreted, as they are represented by dependence and independence relationships, two basic human notions. Their construction is fairly easy, although we should pay some attention in the growth of the model that leads to the exponential growth of the probability matrices.

An issue that deserves further attention is the fact that BBN may not cover all the cases that can appear. This undesirable situation occurs when the parent attributes of the new "unknown " projects have not been met in the training dataset, so the method becomes incapable of making an estimation. It is reasonable to expect that this problem will diminish in large datasets. In any case, BBNs may be used in combination with another estimating method, when incapable of producing an estimate for a specific project. In software cost estimation this is common practice. In addition, BBNs present the usual drawback of all machine learning techniques, i.e. the possibility of over-specialization to the training data. For this reason, the sensitivity of the method to various parameters must be investigated in order to consolidate the obtained results.

Further research needs to focus on confirming and evaluating these results on large multi-organizational datasets, such as those coming from ISBSG. The results in that case will be more indicative of the suitability of the approach. Another interesting idea is to provide a systematic, automated approach to combine expert knowledge and empirical data in order to produce the structure of the model, allowing at the same time the updating of the model.

References

1. L.Angelis, I.Stamelos, A Simulation Tool for Efficient Analogy Based Cost Estimation, Empirical Software Engineering 5,35-68 (2000)
2. B.Boehm, Software Engineering Economics, Prentice-Hall: Englewood Cliffs, N.J, 1981
3. E.Charniak, "Bayesian Networks without Tears", AI Magazine 1991.
4. G.Cooper, E.Herskovits , "A Bayesian Method for the Induction of Probabilistic Networks from Data", Machine Learning 9, 1992
5. A.Heiat, Comparison of artificial neural network and regression models for estimating software development effort, Information and Software Technology 44(2002) pp.911-922.
6. F.Jensen, Bayesian Networks and Decision Graphs, Springer, 2002.
7. M.Jorgensen, An effort prediction interval approach based on the empirical distribution of previous estimation accuracy, Information and Software Technology 45 (2003)123-126.
8. B.Kitchenham, A procedure for analyzing unbalanced datasets, IEEE Transactions on Software Engineering, 24, 4(April 1998),pp278-301.
9. B.Kitchenham , S.Linkman , Estimates, uncertainty and risk, IEEE Software14 (3) ,1997, pp 69-74.

10 K.Maxwell, L.Briand, K.Emam, D.Surmann, I.Wieczorek, An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques, Proceedings of the 22nd International Conference on Software Engineering ,ICSE 2000,Limerick,2000, pp.377-386.

11 M.Neil, N.Fenton, Predicting Software Quality using Bayesian Belief Networks,Proceedings of the 21st Annual Software Engineering Workshop ,1996,

12. J.Pearl, "Causality", Cambridge University Press, 2000.
13. J.Pearl, "Probabilistic Reasoning in Expert Systems :Networks of Plausible Inference.San Mateo, Calif.:Morgan Kaufmann.
14. SERENE Home, http://www.hugin.dk/serene.
15. K.Srinivisan, D.Fisher.Machine learning approaches to estimating software development effort, IEEE Transactions on Software Engineering,21,2 (February 1995),pp. 126-137.

16. I.Stamelos, L.Angelis, P.Dimou, E.Sakellaris, On the use of Bayesian belief networks for the prediction of software productivity, Information and Software Technology 45 (2003) 51-60.
17. I.Stamelos, L.Angelis, Managing uncertainty in project portfolio estimation, Information and Software Technology 43 (13),2001, pp 759-768.
18. H.Sturge , The choice of Class Interval, Journal of American Statistical Association(1926) , pp 65-66.
19. G.Subramanian, An Empirical Examination of Software Development Modes, J.Systems Software.
20. H.Ziv, D.Richardson, Constructing Bayesian-network Models of Software Testing and Maintenance Uncertainties, Proceedings of the International Conference on Software Maintenance,1997.