On generating Bayesian nets from small local qualitative data for software development effort and quality prediction

Łukasz Radliński

University of Szczecin, Department of Information Systems Engineering

Abstract:

The aim of this paper is to investigate if it is possible to build accurate Bayesian net models for software development effort and quality prediction under two assumptions for model generation: (1) no expert knowledge is incorporated, (2) only small local qualitative data is used. Models generated in this study provide predictions with medium level of accuracy, yet still keeping the literature average. Thus, they can be used to make only rough estimations at the early software development stage. However, they can be a useful base for detailed models incorporating expert knowledge and tailored for individual needs.

Keywords:

Bayesian nets, effort prediction, quality prediction, local data, process factors

1. Introduction

Software engineering and statistics literature suggest using large and homogeneous datasets to predict effort and quality. This requirement can hardly be met in industry as extensive metrics programs are expensive. Additionally, many statistical methods require meeting specific assumptions in respect with the data, like normal distribution, linear relationships etc. In industrial use, the analysis of such constraints is often omitted, possibly due to lack of awareness of them. This leads to incorrect use of particular methods. In other cases, managers skip the whole modelling process at all and get "predictions" come purely on intuition.

Thus, there is a need for such modelling method, which is easy to use for non-specialists in statistics, artificial intelligence (AI) or related fields. The procedure of building a model should be flexible about incorporating expert knowledge, generating the model purely from empirical data or mixing expert knowledge with empirical data, depending on particular industrial needs and time available. Such method should also allow presenting both the model and the results in a clear way.

Bayesian net (BN) is one of very few methods, which meet such criteria. This probabilistic method has been successfully used in various studies, including in software engineering field [2], [7], [8], [14], [15], [17], [20], [21], [25], [37], [38], [41], [42] – more details will come in Section 2.

The aim of this study is to analyze if BNs can be effectively used to predict software development effort and software quality. There are two assumptions in this experiment – realistic and important, especially from industrial perspective:

– Available dataset is small but contains local data about past projects from a single company.

– A predictive model is automatically generated from available data without expert input.

One of the most important advantages of BNs is the ability of incorporating expert knowledge and empirical data – with different proportions, depending on particular environment. In BN models that incorporate expert knowledge, this incorporation typically takes place at one of the early stages of model development (when there is no working model yet). This process is time-consuming so it is difficult to follow such approach where there is a pressure to build a model fast. Generating a BN automatically from data, i.e. without expert input, simplifies and shortens the process of building a BN. However, it is at the cost of model correctness (many, even trivial/deterministic relationships may not be discovered) and/or the accuracy of predictions.

The main research question analyzed in this paper is: *can we use BNs automatically generated from small local data to predict effort and quality*? The process of generating such models is easy to follow mainly because it does not require detailed domain-specific or statistical knowledge. It has is a strong practical aspect – as in a typical industrial setting low volume of data and limited methodological knowledge available.

There are many other methods, possibly allowing more accurate predictions, but they require additional knowledge and take more time to build a predictive model. Based on observations of the industrial needs, it appears that there may be a place for methods more accurate but more difficult to use, and for methods less accurate but easy to use. In this paper we focus on the latter by analysing if software companies can make use of such small and relatively cheap datasets to predict effort and quality.

The main contribution of this paper is an analysis in which we have automatically generated a set of BNs from small dataset without user input. This includes the discussion of the features of these BNs. Furthermore, this paper provides a summary of recent BNs for software engineering area built either purely by an expert, automatically from data, or using a mixture of expert knowledge and empirical data. This paper is an extended version of an earlier study [33]. It provides more details on performed experiment and results. Additionally, it contains a sensitivity analysis not discussed earlier.

This paper is organized as follows: Section 2 provides background on BNs. Section 3 contains the description of a dataset and a method used in this study. Achieved results, compared with results from earlier studies, have been discussed in Section 4. Section 5 draws main conclusions.

2. Bayesian nets

2.1. Bayesian nets background

Bayesian net [9], [18], [23], [27], [34] is a probabilistic model which contains two perspectives: graphical and numeric. Graphically BN is a directed acyclic graph consisting of a set of nodes (variables) and directed links between pairs of nodes. Through this graphical representation BNs allow to clearly reflect relationships between variables. Numerically each node is defined in terms of conditional probability distribution given the states of its parent nodes. Thus, the joint probability distribution over its variables X_i, ..., X_n is defined as:

$$\mathrm{P}(X_i,\ldots,X_n)=\prod_{i=1}^{n}\mathrm{P}(X_i \mid \mathrm{Parents}(X_i)). \tag{1}$$

BNs allow both forward and backward reasoning using a Bayes' Theorem [3]:

$$\mathrm{P}(Y \mid X) = \frac{\mathrm{P}(X \mid Y)\,\mathrm{P}(Y)}{\mathrm{P}(X)}, \tag{2}$$

where:

- $P(Y|X)$ is the posterior probability (to be calculated) of event Y given event X, i.e. it incorporates all evidence X,
- $P(Y)$ is the prior probability of event Y,
- $P(X|Y)$ is the likelihood of evidence X given event Y,
- $P(X)$ is the prior probability of event X – it is a normalizing constant which can be computed as: $\mathrm{P}(X) = \sum_j P(X \mid Y_j) P(Y_j)$.

Although Bayes' Theorem was formulated back in the XVIII century, BN concepts (and the term itself) were introduced in the 1980's in pioneering work by Pearl [26], [27], and since then they "have revolutionized AI" [28].

The motivation for selecting BNs in this study is the unique set of their features:

- ability to build models in several ways, depending on available data and domain knowledge – purely from the data, purely by an expert or using a combination of empirical data and expert knowledge,
- ability to reflect causal relationships when the model structure is built by an expert,
- explicit incorporation of uncertainty in the form of probability distributions for model variables,
- ability of both forward and backward reasoning what gives a powerful base for simulation experiments,
- graphical representation of model topology is clear and intuitive.

2.2. Related BN Models

Table 1 summarizes some of recent BNs for software engineering. Most of these models have been built to predict either effort or quality (various measures), some enable an integrated effort and quality prediction. In the very few other models the focus has been set to other aspects, mostly related to requirements specification phase. Four main types of structures have been used in these BNs:

- Naive Bayesian classifier (NBC) – a structure of a diverging star with all the links directed from a single dependent variable to each predictor.
- Converging star (CS) – topology similar to NBC but with all links reversed – pointing from each predictor to a single dependent variable.
- Causal BN (CBN) – contains causal relationships between variables.
- Dynamic BN (DBN) – a sequence of CBNs linked together, where each instance represents system state at specific point of time.

When a model is built purely from the data, a learning algorithm produces a CBN structure. However, it does not ensure creating causal links but rather links, which represent strong statistical dependencies. All of those structures can be used to learn parameters from the dataset. CBNs, DBNs and, partially, NBCs can be used to incorporate expert knowledge in parameter definition.

A general problem related with developing BNs is their validation. It is often expected to validate the predictive model it terms of accuracy of its predictions. To do this testing data sample is required. However, BNs are often selected as a modeling technique because there is no data of required volume/quality available to build the model purely from the da-

ta. As a result of this discrepancy many analyzed BNs have not been validated for their accuracy. In some studies where such validation has been performed, the testing data sample was very small and comparing their performance with other models may be misleading. Among all the BNs summarized in Table 1 only five models from [14], [15] and [20], [21] can be directly compared with the current study. We have performed detailed analysis of earlier BN models for software effort prediction in recent paper [29].

Table 1. Summary of recent BNs for software engineering

Source	**Main problem analyzed**	**Model type**	**Structure source**	**Parameter source**	**Validated with empirical data**
[32]	types of defects	NBC	expert	data/expert	no
[37]	effort, productivity	NBC	expert	data	yes
[38]	maintainability	NBC	expert	data	yes
[25]	fault content, fault proneness	CS	expert	data	yes
[42]	change coupling	CS	data	data	yes
[10]	maintenance delays	CBN	expert	expert/data	no
[11]	need for requirements review	CBN	expert	expert	no
[13]	trade-off between: scope, effort, quality	CBN	expert	expert/data	no
[14], [15]	defects, partly: effort	CBN/ DBN	expert	expert/data	yes
[20], [21]	web development effort	CBN	expert/data	expert/data	yes
[24]	maturity of requirements	CBN	expert	expert/data	no
[30], [31]	trade-off between: scope, effort, quality	CBN	expert	expert/data	no
[39]	various aspects of software quality	CBN	expert	expert/data	no
[41]	testing process	CBN	expert	expert/data	yes
[7]	effectiveness of inspections	CBN	expert	expert	yes
[8]	defect rate	CBN	expert	data/expert	yes
[2]	failures	DBN	expert	data	yes
[4]	effort	DBN	expert	*unknown*	no
[12], [31]	defects	DBN	expert	expert	no
[17]	project velocity (productivity)	DBN	expert	expert/data	yes

3. Material and Method

3.1. Material

In this experiment, we have used an extended version of publicly available dataset [5], [14] of 31 software projects from a single company. This extension includes one additional predictor – *project type*. We have investigated four dependent variables:

- *effort* – in person-hours,
- *productivity rate* (a ratio of *effort* to *project size*) – in thousands lines of code (KLoC) per hour,
- *number of defects*,
- *defect rate* (a ratio of *number of defects* to *project size*) – as number of defects per KLoC.

The dataset contains 29 predictors: project size, project type and 27 factors describing process and people quality expressed on a 5-point ranked scale (from 'very low' to 'very high'). Table 2 lists all variables.

We have excluded two cases from the original dataset because they did not contain data on *project size*, which is one of the main factors in effort and defect prediction. Additionally, it would not be possible to calculate productivity and defect rates for these projects. Thus finally, we have used 29 cases in main analysis.

Table 2. Variables used in this study

Name	Role[1]	Type[2]	Name	Role[1]	Type[2]
complexity	P	R	project size	P	NU
configuration management	P	R	project type	P	NO
customer involvement	P	R	quality of doc. test cases	P	R
defect rate	D	NU	quality of documentation	P	R
defects	D	NU	regularity of spec&doc reviews	P	R
defined process followed	P	R	requirements stability	P	R
dev. staff experience	P	R	review process effectiveness	P	R
dev. staff motivation	P	R	scale of distr. comm.	P	R
dev. staff training quality	P	R	scale of new functionality	P	R
effort	D	NU	spec. defects discovered in review	P	R
experience of spec&doc staff	P	R	staff experience - indep. test	P	R
internal comm./interaction	P	R	staff experience - unit test	P	R
no. of inputs and outputs	P	R	stakeholder involvement	P	R
process maturity	P	R	standard procedures followed	P	R
productivity	D	NU	testing proc. well defined	P	R
programmer capability	P	R	vendor management	P	R
project planning	P	R			

[1] D – dependent variable, P – predictor

[2] NO – Nominal, NU – Numeric, R – Ranked

3.2. Research Method

The research procedure followed in this study consists of various steps as illustrated in Figure 1. Data preparation involved adjusting categories for project type to ensure at least five cases for each category. Then, we have discretized each numeric variable (as required by a learning algorithm) into four or five categories.

Then we have randomly split the dataset into training and testing subsets. Since there is no standard for splitting proportions, in this study we have used about 75% (22 cases) for model generation and about 25% (7 cases) for validation. In the literature more common are 67%–33% and 50%–50% splits. However, with such small dataset there was a danger that with using even fewer cases for model generation, the predictions would be highly inaccurate and noisy.

The process of model generation involved both the structure and parameter (probability distribution) learning using Greedy Thick Thinning algorithm implemented in Genie [16]. At this stage, we have replaced a couple of missing values for predictors by an average value of given predictor in the learning subset.

Then we have imported the model to another BN tool, AgenaRisk [1], to perform the validation step. Although this could also have been performed in Genie, AgenaRisk has been selected because it provides summary statistics for continuous nodes and because of

the ability to use yet another tool (developed earlier) based on Agenerisk API, which simplifies batch model testing. The testing phase was a single step, which involved providing predictions for all four dependent variables at once using a set of predictors. It means that *defects* and *defect rate* have been predicted without the use of observed *effort* and *productivity rates*.

Based on experience with developing earlier models [12], [14], [31] the median of the posterior distribution has been selected as the "predicted value" for each dependent variable. Experiments with earlier models have shown that the mean of predicted probability distribution is less accurate for effort and quality prediction. This is because, typically, these distributions are heavily right-skewed, and the mean value is "pushed" more to the right.

To ensure that the results are not based on a single dataset split, all the steps after data preparation have been repeated 10 times – for different random splits of data.

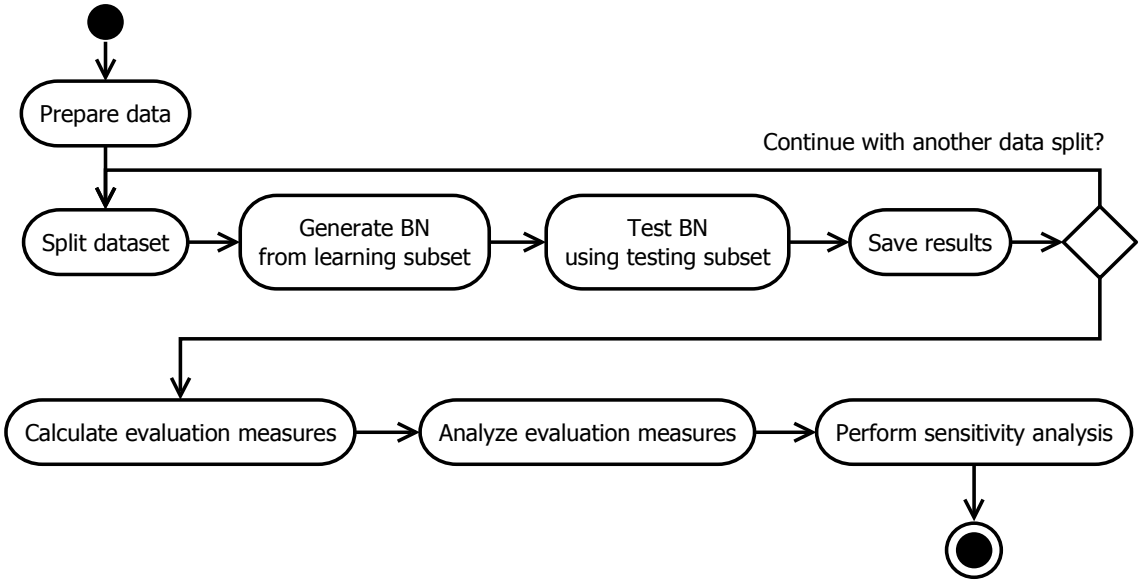

Figure 1. Research procedure

The analysis of results involved calculating measures of predictive accuracy. In this study the following popular evaluation measures have been applied:

– MMRE (mean magnitude of relative error):

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i \,, \tag{3}$$

$$MRE_i = \left| \frac{y_i - \hat{y}_i}{y_i} \right| . \tag{4}$$

– MdMRE (median magnitude of relative error):

$$MdMRE = \text{Median}(MRE_i) \tag{5}$$

– Pred(l) (prediction at level l) – indicating the fraction of cases for which predictions are within ±*l* percent of actuals:

$$Pred(l) = \frac{1}{n}\sum_{i=1}^{n} a_i \,, \tag{6}$$

$$a_i = \begin{cases} 1 \ \ if \ \ MRE_i \le \dfrac{l}{100} \\ 0 \ \ if \ \ MRE_i > \dfrac{l}{100} \end{cases}. \tag{7}$$

In different studies, the authors have also used other measures like balanced mean magnitude of relative error or mean/median magnitude of relative error relative to the estimate [19], [36]. However, their interpretation is less intuitive and we decided not to use them.

Apart from analyzing predictive accuracy of developed models, this study involved identification of factors closely related with the dependent variables. We have done this by analyzing Markov blankets for generated BNs. A Markov blanket for a node *A* is a set of nodes which, if they are instantiated (i.e. observations have been assigned to them), are the only nodes influencing node *A*. Thus, it shields a node from the impact of nodes not belonging to its Markov blanket. It can be identified from the BN structure – for a node *A* it contains:

- parent nodes for *A*,
- child nodes for *A*,
- other parent nodes for child nodes for *A*.

For each of ten developed BNs, we have identified the most frequent predictors in Markov blanket for each dependent variable. We have also focused on indentifying the most important predictors using another technique – a sensitivity analysis.

4. Results

4.1. Accuracy of predictions

Figure 2 illustrates the values of evaluation measures from each of ten individual samples/BNs. The values of these measures differ significantly depending on the data sample used. This shows that in some samples projects in testing subset have been significantly different from projects in training subset. Thus, the patterns for projects in training subsets sometimes have not been learnt to a degree that enables accurate predictions for projects using testing subsets. A possible reason for this is a high diversity of projects, underrepresented in the dataset, even though they have been developed in a single company.

Table 3 summarizes the predictions by demonstrating different evaluation measures. Results of this study have been compared with two earlier studies where BNs have been used for:

- Defect prediction [14] – the authors built the BN structure and defined parameters prior to obtaining the dataset (the same dataset as in this study); thus, they used the dataset only in model validation;
- Effort prediction for web development [20] – the authors built various BNs, and then validated them against two datasets, each containing data on 65 projects.

Predictions from BNs generated automatically without expert input do not seem to be highly accurate. However, effort and quality prediction is generally a demanding task – high predictive accuracy is very rare in the literature, also where other methods have been used. This comparison reveals that predictions in this study are respectable – worse than in study [14] for defect prediction but significantly better than study [20] for effort prediction. They keep the average accuracy in the literature. Yet, the process of achieving them, through automatic BN generation, is easy to follow for non-statisticians and free from various constraints.

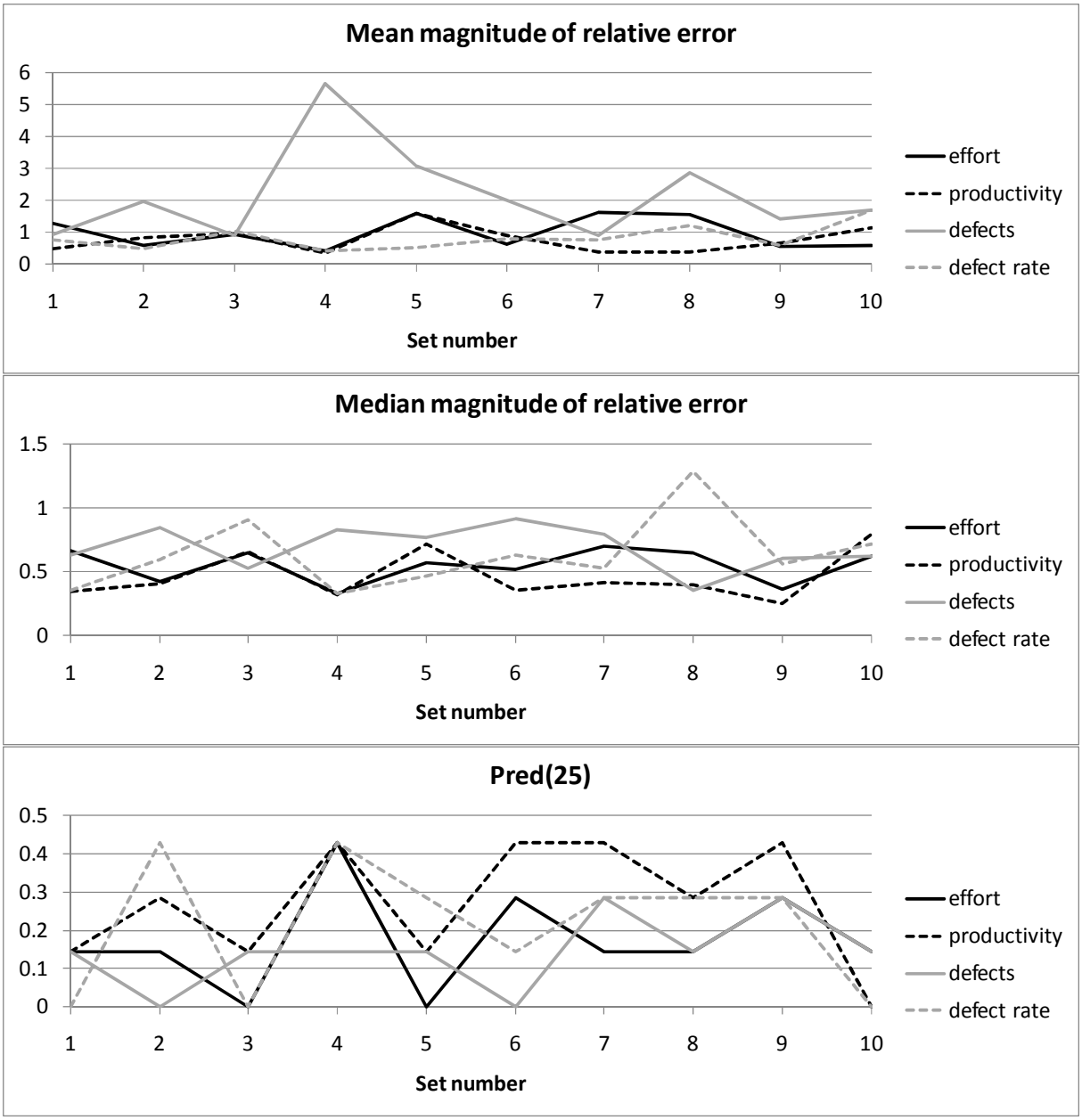

Figure 2. Values of evaluation measures from individual samples/BNs

Additionally, it should be noted that all four dependent variables have been included in the model not in their natural continuous form, but each of them discretized into four intervals. Such reduced precision could have a negative impact on limited accuracy. Furthermore, we have also deliberately discretized project size, the most important predictor for effort and defects in many models. The motivation for discretizing project size, apart from such requirement by a structure learning algorithm, was that in real projects this size can never be precisely known at the early development stage.

Table 3. Summary of predictions

Study and dependent variable	Evaluation measure		
	MMRE	**MdMRE**	**Pred(25)**
this study:			
effort	0.97	0.60	0.17
productivity	0.77	0.40	0.27
defects	2.14	0.79	0.14
defect rate	0.83	0.53	0.21
in [14]:			
defects	0.96	0.27	0.58*
in [20]: dataset 1			
effort (1)	13.97	2.57	0.05
effort (2)	7.65	1.67	0.08
effort (3)	36.00	4.90	0.08
effort (4)	1.90	0.86	0.15
in [20]: dataset 2			
effort (1)	14.93	6.46	0
effort (2)	4.09	0.96	0.02
effort (3)	37.31	8.05	0.02
effort (4)	27.95	5.31	0.03

* Pred(30)

4.2. Most important predictors

Figure 3 illustrates parts of the structures of two best performing BNs. We have nominated the best performing BNs based on the measure of performance score for each model (lower the better) defined as:

$$Performance_j = \frac{\sum_{i=1}^{4} MMRE_{ij}}{\sum_{j=1}^{10}\sum_{i=1}^{4} MMRE_{ij}}, \tag{8}$$

where i denotes an index for dependent variable (i = 1, ..., 4) and j denotes an index for the model generated (j = 1, ..., 10). Although, some generated BNs performed better for a single dependent variable, these two BNs performed the best overall, i.e. on average with all dependent variables. To improve the figure clarity, it only contains those variables, which are within a Markov blanket for at least one dependent variable.

These structures are generally different. There are some similarities, for example the dependencies for *defects*, and partially for *effort*. However, there are many links in one model that do not appear in the second. This again confirms the variability of projects, which are not sufficiently represented in the dataset.

Two dependent variables (*productivity rate* and *defect rate*) are defined deterministically as the ratio of *effort* and *defects*, respectively, with *project size*. Unexpectedly, a learning algorithm has not identified these relationships in the generated BNs, apart from some partial exceptions, like relationship between *productivity rate* and *project size*. It may suggest that such trivial relationships should rather be identified by experts and encoded in the model, before learning the main structure from the dataset.

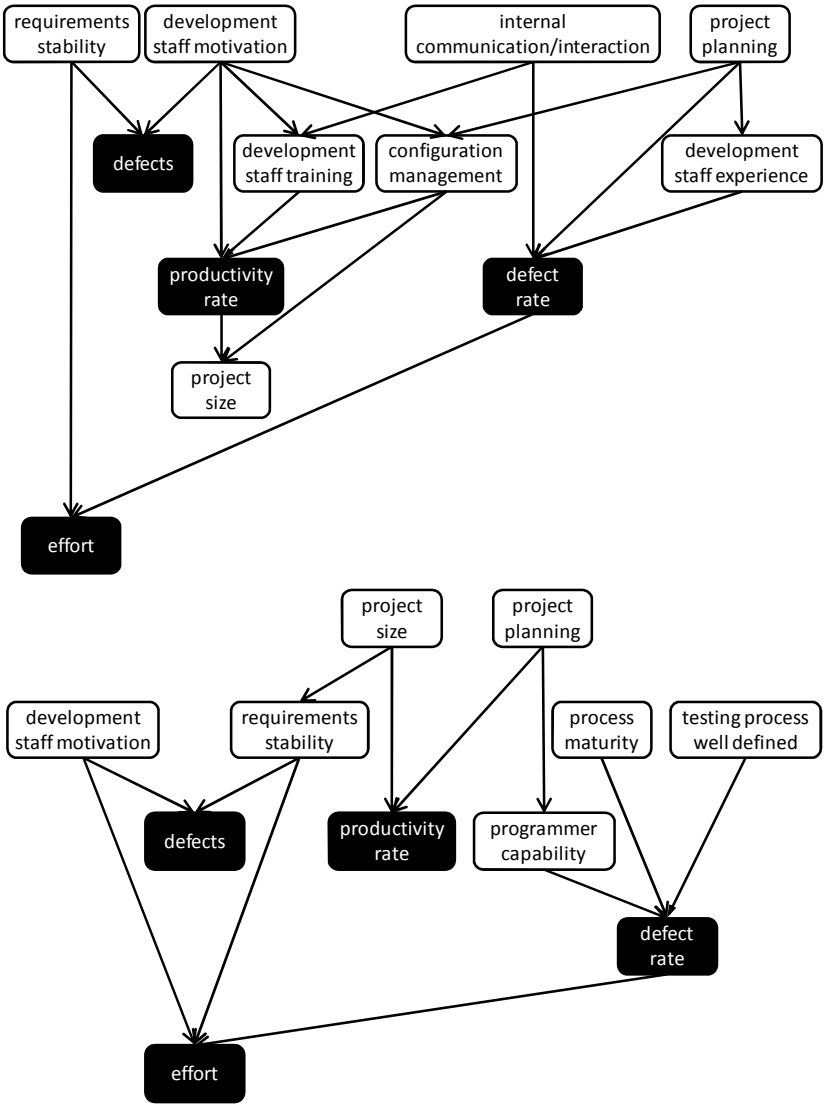

Figure 3. Schematics of the cores of two best performing BNs:
from subset no. 9 (top) and subset no. 1 (bottom)

Table 4 contains the counts – number of times that each predictor appears in the Markov blanket for each dependent variable. This table also lists dependent variables in rows (i.e. as predictors) because in some learned BNs they have been in Markov blanket for other dependent variable – for example *defect rate* as predictor for *effort*. The other predictors (i.e. real ones) have been listed in the order of the decreasing total frequency in Markov blankets in all BNs.

Three predictors (*development staff motivation*, *project size* and *requirements stability*) have been the most frequent predictors for dependent variables. Surprisingly, *project size* is a dominating predictor only for *productivity rate*, and not for *effort* or *number of defects*. This can be explained by the fact that the variability of *project size* end *effort* was generally low in the dataset.

A variable *number of defects* seem to have strong predictors identified – *requirements stability* appeared in each BN's Markov blanket and *development staff motivation* appeared in eight of ten Markov blankets. No other dependent variable had such dominating predictors. Yet, it is the *number of defects*, which had the lowest prediction accuracy among all

four dependent variables. The explanation for this might be that, although these predictors have been the related to *number of defects* so often, there might be generally low level of correlation between these predictors and *defects*. This confirms the general difficulty of defect prediction.

Table 4. Frequencies of factors appearing in Markov blankets for predictive variables

Factor	**Dependent variable**			
	effort	**productivity rate**	**defects**	**defect rate**
effort	–	0	1	**6**
productivity	0	–	3	0
defects	1	3	–	0
defect rate	**6**	0	0	–
dev. staff motivation	**7**	1	**8**	4
project size	**5**	**8**	2	2
requirements stability	**5**	0	**10**	1
complexity	3	0	**6**	0
project planning	1	4	0	1
dev. staff experience	0	2	2	2
vendor management	4	0	0	2
testing proc. well defined	2	1	0	2
staff experience - unit test	2	0	0	3
programmer capability	0	1	0	3
scale of distr. comm.	1	0	3	0
configuration management	0	4	0	0
internal comm./interaction	0	1	2	1
process maturity	0	2	0	1
dev. staff training quality	0	3	0	0
experience of spec&doc staff	1	0	0	1
customer involvement	0	0	0	2
stakeholder involvement	1	1	0	0
spec. defects discovered in review	1	1	0	0
project type	0	1	0	0
regularity of spec&doc reviews	0	0	1	0
scale of new functionality	0	0	0	1
no. of inputs and outputs	0	0	1	0
staff experience - indep. test	0	0	0	1
defined process followed	1	0	0	0

To investigate the impact of the most important predictors on dependent variables we have performed a sensitivity analysis. There are various types of sensitivity analyses [6], [14], [22], [35], [40]. In this study, we have analyzed the strength of impact of each predictor on each dependent variable in the best overall performing BN. For each predictor we have entered observations for each possible state and then we have calculated the model. In each calculation run, we have entered only one observation into the model. For each dependent variable we have calculated the range of changes caused by observations for predictors. Figure 4 illustrates these ranges. The wider bar for a predictor indicates its proportionally higher impact on dependent variable than for other predictors.

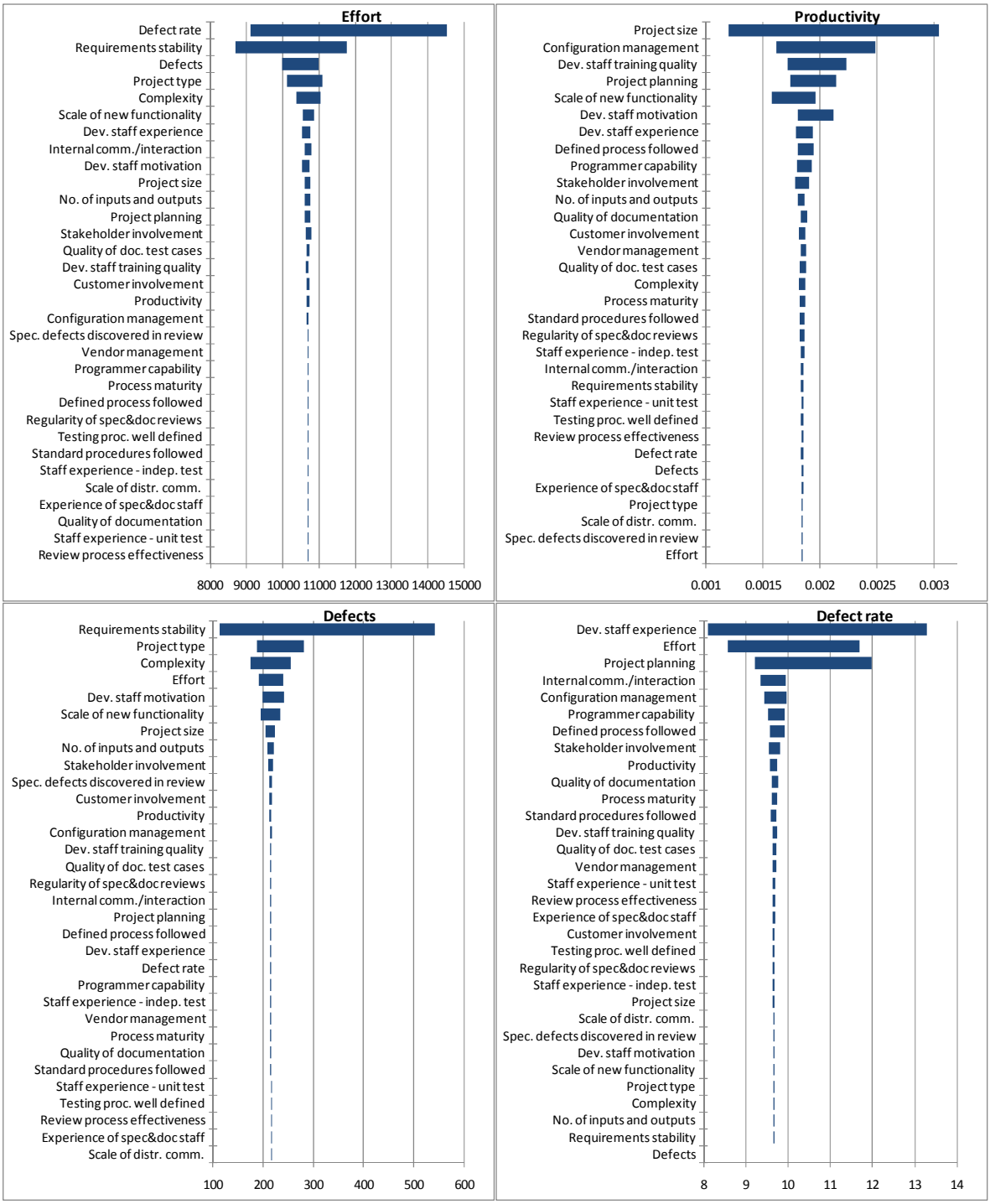

Figure 4. Results of sensitivity analysis from best performing BN for each dependent variable

Some predictors appear to have a high impact on specific dependent variables, although these predictors very rarely or never appeared in the Markov blankets discussed earlier. For example, *project type*, which is the fourth most important predictor for effort and second for defects, does not appear in Markov blanket in the analyzed best performing BN and appeared in only one Markov blanket in ten generated models. Such situations occur because there are strong correlations between dependent variable and a predictor within a Markov blanket, and between this predictor and yet another one from outside of Markov blanket. Thus, predictors from outside Markov blanket reveal their impact on dependent variable

only when predictor within Markov blanket are not instantiated, i.e. do not have observation assigned. It shows that a single method for determining most influential predictors may not reflect the reality properly.

We note that the ends of ranges do not indicate the minimum and maximum values for dependent variables. Predictions for dependent variables are in the form of probability distributions but possible ranges are fixed for each dependent variable. As explained earlier, also in this sensitivity analysis we have used the calculated median of the probability distribution as the "predicted value".

5. Conclusions

The research method with automatic generation of BNs from empirical data followed in this study is easy to use – it does not involve significant data preparation or excessive preliminary analyses as in some other methods. The low volume of data used in model generation does not allow to exploit these BNs in other environments. Nevertheless, the relationships identified during model generation can still be a useful base for further studies.

The results of this experiment have shown that it is difficult to obtain highly accurate predictions from BNs built only from small empirical data. Thus, such automatically generated model can only be used as a "rough" estimate of the variable under consideration at the beginning of the project. They can also serve as general maps of relationships for further experiments.

There are two general approaches, which can be followed to solve the problem of low accuracy – both of them require stronger input from domain expert. The first approach assumes that an expert prepares a BN topology while the parameters can be learnt automatically from data. Thus, this approach enables defining relationships between the variables, which might have not been discovered by a structure learning algorithm.

In the second approach, the BN structure and parameters are still learnt automatically from available data, but then an expert may adjust parameter definition. This enables solving the problem arising from the fact that such small local dataset may not be representative.

Future plans for this research involve investigating the predictive accuracy of other methods using the same dataset and similar procedure, where possible. Specifically, this includes using another graphical technique, decision trees, generated using various algorithms. Also, part of the current study may be repeated in different setting, with effort and productivity assumed to be known, to simulate defect prediction at the end of development stage.

Acknowledgement

I would like to thank Professor Marek Druzdzel (University of Pittsburgh, USA) and Professor Norman Fenton (Queen Mary, University of London, UK) for providing tools for developing and managing BNs: Genie and AgenaRisk, respectively.

References

[1] *AgenaRisk*. Agena, Ltd., 2009 [online] *http://www.agenarisk.com* [accessed: 2011]

[2] Bai C.G., Hu Q.P., Xie M., Ng S.H. *Software failure prediction based on a Markov Bayesian network model*. Journal of Systems and Software, Vol. 74(3), 2005, pp. 275-282

[3] Bayes T. *An essay towards solving a Problem in the Doctrine of Chances. By the late Rev. Mr. Bayes, communicated by Mr. Price, in a letter to John Canton, M.A. and F.R.S.* Philosophical Transactions of the Royal Society London, Vol. 53, 1763, pp. 370-418
[4] Bibi S., Stamelos I. *Software Process Modeling with Bayesian Belief Networks*. Proc. of the International Software Metrics Symposium, Chicago, 2004
[5] Boetticher G., Menzies T., Ostrand T. *PROMISE Repository of Empirical Software Engineering Data*, West Virginia University, Department of Computer Science, 2010 [online] *http://promisedata.org/repository* [accessed: 2010]
[6] Cangussu J.W., DeCarlo R.A., Mathur A.P. *Using sensitivity analysis to validate a state variable model of the software test process*. IEEE Transactions on Software Engineering, Vol. 29(5), 2003, pp. 430–443
[7] Cockram T. *Gaining Confidence in Software Inspection Using a Bayesian Belief Model*. Software Quality Journal, Vol. 9(1), 2001, pp. 31–42.
[8] Dabney J.B., Barber G., Ohi D. *Predicting Software Defect Function Point Ratios Using a Bayesian Belief Network*. Proc. of the 2nd International Workshop on Predictor Models in Software Engineering, Philadelphia, PA, 2006
[9] Darwiche A. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009
[10] de Melo A.C.V., Sanchez A.J. *Software maintenance project delays prediction using Bayesian Networks*. Expert Systems with Applications, Vol. 34, 2008, pp. 908-919
[11] del Salgado Martinez J., del Aguina Cano I.M. *A Bayesian Network for Predicting the Need for a Requirements Review*. [in:] Meziane F., Vadera S. (eds.) *Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects.* Information Science Reference, New York, 2008, pp. 106-128
[12] Fenton N., Hearty P., Neil M., Radliński Ł. *Software Project and Quality Modelling Using Bayesian Networks*. [in:] Meziane F., Vadera S. (eds.) *Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects.* Information Science Reference, New York, 2008, pp. 1-25
[13] Fenton N., Marsh W., Neil M., Cates P., Forey S., Tailor M. *Making Resource Decisions for Software Projects*. Proc. of the 26th International Conference on Software Engineering, Washington, DC, IEEE Computer Society, 2004, pp. 397–406
[14] Fenton N., Neil M., Marsh W., Hearty P., Radliński Ł., Krause P. *On the effectiveness of early life cycle defect prediction with Bayesian Nets.* Empirical Software Engineering, Vol. 13, 2008, pp. 499–537
[15] Fenton N.E., Neil M.., Marsh W., Krause P., Mishra R. *Predicting Software Defects in Varying Development Lifecycles using Bayesian Nets.* Information and Software Technology, Vol. 43(1), 2007, pp. 32–43
[16] *Genie*. Decision Systems Laboratory, University of Pittsburgh, 2009 [online] *http://genie.sis.pitt.edu* [accessed: 2010]
[17] Hearty P., Fenton N., Marquez D., Neil M. *Predicting Project Velocity in XP using a Learning Dynamic Bayesian Network Model*. IEEE Transactions on Software Engineering, Vol. 37(1), 2009, pp. 124–137
[18] Jensen F.V. *An Introduction to Bayesian Networks*. UCL Press, London, 1996
[19] Kitchenham B.A., Pickard L.M., MacDonell S.G., Shepperd M.J. *What accuracy statistics really measure*. IEE Proceedings Software, Vol. 148(3), 2001, pp. 81–85
[20] Mendes E., Mosley N. *Bayesian Network Models for Web Effort Prediction: A Comparative Study*. IEEE Transactions on Software Engineering, Vol. 34, 2008, pp. 723-737

[21] Mendes E. *The Use of Bayesian Networks for Web Effort Estimation: Further Investigation*. Proc. of the 8th International Conference on Web Engineering, Yorktown Heights, NJ, 2008, pp. 203-216
[22] Musilek P., Pedrycz W., Nan Sun, Succi G. *On the Sensitivity of COCOMO II Software Cost Model*. Proc. of the 8th IEEE Symposium on Software Metrics, 2002, pp. 13–20
[23] Neapolitan R.E., *Learning Bayesian Networks*. Pearson Prentice Hall, Upper Saddle River, 2004
[24] Pai G., Bechta-Dugan J., Lateef K. *Bayesian Networks applied to Software IV&V*. Proc. of the 29th Annual IEEE/NASA Software Engineering Workshop, IEEE Computer Society, Washington, DC, 2005, pp. 293-304
[25] Pai G.I., Dugan J.B. *Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods*. IEEE Transactions on Software Engineering, Vol. 33(10), 2007, pp. 675-686
[26] Pearl J. *Bayesian Networks: A Model of Self-Activated Memory for Evidential Reasoning*. (UCLA Technical Report CSD-850017). Proc. of the 7th Conference of the Cognitive Science Society, University of California, Irvine, 1985, pp. 329–334
[27] Pearl J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, 1988
[28] Poole D., *Review of Modeling and Reasoning with Bayesian Networks*. 2009 [online] http://www.cambridge.org/us/catalogue/catalogue.asp?isbn=9780521884389 [accessed: 2011]
[29] Radlinski L. *A Survey of Bayesian Net Models for Software Development Effort Prediction*. International Journal of Software Engineering and Computing, Vol. 2(2), 2010, pp. 95-109
[30] Radliński Ł., Fenton, N., Neil, M., Marquez D. *Improved Decision-Making for Software Managers Using Bayesian Network.* Proc. of the 11th IASTED International Conference Software Engineering and Applications, Cambridge, MA, 2007, pp. 13–19
[31] Radlinski L. *Improved Software Project Risk Assessment Using Bayesian Nets*. Ph.D. Thesis, Queen Mary, University of London, London, 2008
[32] Radliński Ł. *Predicting Defect Types in Software Projects*. Polish Journal of Environmental Studies, Vol. 18(3B), 2009, pp. 311-315
[33] Radliński Ł. *Software Development Effort and Quality Prediction Using Bayesian Nets and Small Local Qualitative Data*. Proc. of the 22nd International Conference on Software Engineering and Knowledge Engineering, Redwood City, CA, 2010, pp. 113-116
[34] Russell S., Norvig P. *Artificial Intelligence. A Modern Approach. Second Edition*. Pearson Education, Inc., Upper Saddle River, 2003
[35] Saltelli A., Chan K., Scott E.M. (eds.) *Sensitivity Analysis*. John Wiley & Sons, 2000
[36] Stensrud E., Foss T., Kitchenham B., Myrtveit I. *An Empirical Validation of the Relationship Between the Magnitude of Relative Error and Project Size*. Proc. of the 8th IEEE Symposium on Software Metrics, 2002, pp. 3–12
[37] Stewart B. *Predicting project delivery rates using the Naive–Bayes classifier*. Journal of Software Maintenance and Evolution: Research and Practice, Vol. 14, 2002, pp. 161–179
[38] van Koten C., Gray A.R. *An application of Bayesian network for predicting object-oriented software maintainability*. Information and Software Technology, Vol. 48, 2006, pp. 59-67
[39] Wagner S. *A Bayesian network approach to assess and predict software quality using activity-based quality models*. Proc. of the 5th International Conference on Predictor Models in Software Engineering, New York, ACM Press, 2009

[40] Wagner S. *Global Sensitivity Analysis of Predictor Models in Software Engineering*, Proc. of the 3rd International Workshop on Predictor Models in Software Engineering, International Conference on Software Engineering, IEEE Computer Society, Washington, DC, 2007, p. 3

[41] Wooff D.A., Goldstein M., Coolen F.P.A. *Bayesian Graphical Models for Software Testing*. IEEE Transactions on Software Engineering, Vol. 28(5), 2002, pp. 510–525

[42] Zhou Y., Würsch M., Giger E., Gall H.C., Lü J. *A Bayesian Network Based Approach for Change Coupling Prediction*. Proc. of the 15th Working Conference on Reverse Engineering, Washington, DC, IEEE Computer Society, 2008, pp. 27–36