INTERNATIONAL JOURNAL OF SOFTWARE ENGINEERING AND COMPUTING
© International Science Press, ISSN: 2229-7413

A SURVEY OF BAYESIAN NET MODELS FOR SOFTWARE DEVELOPMENT EFFORT PREDICTION

Lukasz Radlinski

This paper discusses recent Bayesian nets built for software development effort prediction. Its aim is to bring closer these models as they may be competitive for other modeling techniques, especially for data-driven machine learning and statistical techniques. Each model has been briefly described and then analyzed in detail in terms of its main purpose, type of structure, data/knowledge base for building a model. Some models have been empirically validated for predictive accuracy and we discuss the results of this validation. The paper also discusses main problems related to building such models by domain experts.

Keywords: *Bayesian net, software engineering, effort prediction, survey, expert knowledge*

1. BACKGROUND

Predicting software development effort has been one of the most important and widely studied areas of software engineering and project management. Various statistical, machine learning and artificial intelligence methods have been investigated to build predictive models. There is no technique that provides the most accurate results for all possible environments. Several studies provide guidelines on selecting the most appropriate technique for a specific problem in software engineering area under various constraints and data availability and quality [7, 8, 22, 27, 50, 59].

Most techniques are heavily based on the availability of empirical data of appropriate volume and granularity. Often software companies do not have such data and thus they can use such techniques in a very limited way. One of the solutions to overcome this problem is to gather more data on development effort. However, this process may be too costly and inefficient. The second solution is to incorporate expert knowledge into predictive models. Bayesian net (BN) is one of techniques that enable this.

More precisely, BNs can be built either purely based on expert judgment, purely based on empirical data or using a combination of both expert judgment and empirical data. This flexibility is a very useful feature that has attracted researchers from diverse fields, including software engineering and project management. BNs also have a set of other advantages such as ability to reflect causal relationships, explicit incorporation of uncertainty as probability distribution for each variable, graphical representation that makes a model clear, ability of both forward and backward inference, and ability to run a model with missing data.

This paper discusses the most relevant BN models that have been built for software development effort prediction. Tha main goal of this paper is to bring closer the technique of BNs by showing how it has been used so far for this problem and what results have the authors achieved.

The paper is targeted for managers and software engineers who do not necessarily have a strong background in statistics and artificial intelligence. Thus, it does not provide technical details in the form of formal equations. Rather, it is focused on the practical issues related with model building and using. It may serve as a starting point for researchers who are new to BNs but would like to see how they perform in software effort prediction. It may also be a reference material for comparing the existing models with those, which will be developed in future.

* Institute of Information Technology in Management, Faculty of Economics and Management, University of Szczecin, Poland, *E-mail: lukrad@uoo.univ.szczecin.pl*

This paper is an extended version of [40]. This extension involves:

- more models used in analysis, especially those which have been built and published between years 2006 and 2010;
- more detailed analysis, specifically related to various classifications of these models, the empirical base used in the process of building the models, and the results of validation of these models;
- the discussion on general problems related to building, validating and using such models in practice.

This paper is organized as follows: Section 2 brings closer a research approach followed in this study. Then, Section 3 provides a brief introduction to Bayesian nets. Section 4 contains a textual overview of analyzed models. These models have been compared in details in Section 5. In Section 6 we discuss some of the most important limitations related to building and using such models by domain experts.

2. RESEACH APPROACH

Finding relevant papers involved searching in popular databases of scientific publications such as IEEE Xplore, ACM Digital Library, Elsevier's ScienceDirect and SpringerLink. We found additional relevant papers by analyzing references and citations of those found initially. To find other papers of these authors this search also involved authors' websites. All of these papers have been analyzed for relevance based on their titles and abstracts.

During this search we found several other papers on Bayesian nets but we kept them out from further investigation. Among them, there were papers on Bayesian net theory. Some of these papers contained some models presented only as a proof of concept for the theoretical work – no practical benefit can be gained from such model. Other papers included models for various applications outside the main area of software development effort prediction. They have also been excluded from further analysis.

After this preliminary search 23 publications have been included in detailed analysis. This analysis involved such issues as the main problem/area modeled, type of model structure, base for building a model (data, knowledge or both), performing empirical validation, and the results of this validation.

The authors of some models also reported issues related to the process of model development and use in target environment. These reports, together with other papers discussing topics on modeling and using BNs, also in other fields such as medicine, have been used to formulate the important limitations of using BNs in practice.

3. INTRODUCTION TO BAYESIAN NETS

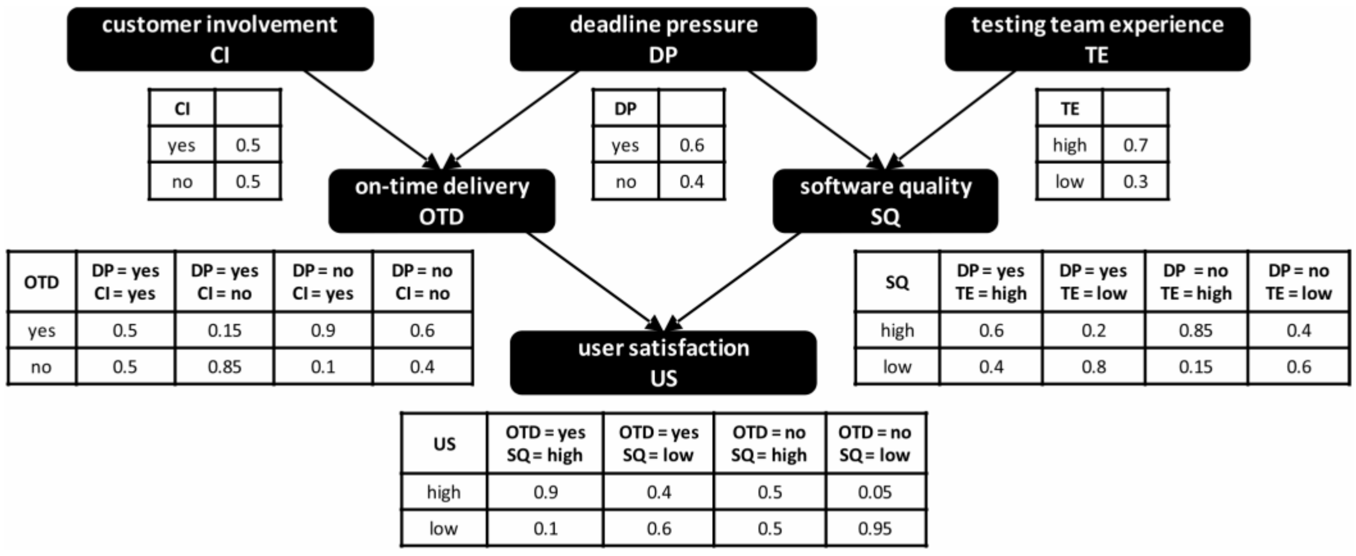
Bayesian net (BN) [12, 25, 37, 47] is a probabilistic model containing a set of random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ that has two perspectives: graphical and numeric. Graphically, it is a directed acyclic graph where a node represents a variable X_i and an arc represents a relationship between pair of variables. The graph structure reflects a set of conditional independence assertions.

Numerically, a BN consists of a set of conditional probability distributions $P(X_i\,|\,parents(X_i))$. Here, $parents(X_i)$ denotes a set of variables being direct predecessors (*parents*) for variable X_i. Thus, each probability distribution encodes the information on the local relationships by defining how variable X_i depends on the states of its parents. Formally, a joint probability distribution for a BN is defined as:

$$P(X_1,\ldots,X_n) = \prod_{i=1}^{n} P(X_i \,|\, parents(X_i))\,. \qquad (1)$$

Figure **1** illustrates an example of a BN for software project management. To keep the probability distributions compact all nodes are binary, i.e. they have only two possible states. Three nodes in the top row (CI, DP and TE) do not have parents and thus are called *root* nodes. Other nodes are called *child* nodes.

The root nodes are not defined conditionally on other nodes – their prior probabilities seem to be unconditional. However, these probabilities are also based on some assumptions and reflect expert belief, the frequency of occurrence in the past or other source of information. Therefore, they are still conditional on these assumptions. For simplicity



CI	
yes	0.5
no	0.5

DP	
yes	0.6
no	0.4

TE	
high	0.7
low	0.3

OTD	DP = yes CI = yes	DP = yes CI = no	DP = no CI = yes	DP = no CI = no
yes	0.5	0.15	0.9	0.6
no	0.5	0.85	0.1	0.4

SQ	DP = yes TE = high	DP = yes TE = low	DP = no TE = high	DP = no TE = low
high	0.6	0.2	0.85	0.4
low	0.4	0.8	0.15	0.6

US	OTD = yes SQ = high	OTD = yes SQ = low	OTD = no SQ = high	OTD = no SQ = low
high	0.9	0.4	0.5	0.05
low	0.1	0.6	0.5	0.95

Figure 1: Example of a Simple BN

their probabilities are written as $P(X_i)$ instead of $P(X_i \mid expert\ knowledge)$, $P(X_i \mid past\ data)$, or $P(X_i \mid other\ available\ information)$.

The process of calculation of a BN model involves using so called Bayes' theorem [3], where for an uncertain variable X and data D:

$$P(X \mid D) = \frac{P(D \mid X)P(X)}{P(D)}. \qquad (2)$$

Calculating complex BN is not a trivial task. Hence, several exact and approximate algorithms have been developed. The discussion of their details is beyond the scope of this paper and interested readers can follow this topic in [12, 25, 37, 47].

Decision network (DN), also called an influence diagram, is a generalization of a BN useful in optimization problems. Apart from random variables, as in a BN, it also contains decision and utility nodes. Decision node reflects a set of possible alternatives that a decision maker may take for a given problem. Utility (or *value*) node reflects how desirable are the combinations of outcomes from the other types of variables, i.e. random and decision variables. Frequently, this utility is expressed in monetary units.

BNs can have various topologies, although these analyzed in this paper fall into four categories summarized in Table 1 and illustrated in Figure 2. The ability to reflect causal relationships is of the greatest advantages of CBN and DBN over other topologies. Both of them may contain multiple dependent variables. Further, there is no fixed list of such dependent variables. During model usage, if a user enters and observation to a variable, this variable becomes a predictor used to calculate posterior probabilities for all other variables, i.e. dependent variables. Additionally, only DBN may intuitively reflect the dynamics of a modeled system. Clear advantages of NB and CS are their simplicity and fast model building, both by an expert and by a learning algorithm. Decision networks typically have structures of CBNs and DBNs, with additional types of variables.

BN models may also be built with other topologies, for example as a Tree Augmented Network, a kind of Naïve Bayes model with additional links among predictors. These additional topologies have not been used in the analyzed models. Thus, we do not discuss them here and we refer to the literature for details on them [9, 19].

4. SUMMARY OF RELATED MODELS

This section provides a general overview of the analyzed BNs for software effort prediction. Figures 3 and 4 summarize analyzed models by illustrating the number of publications that present related BNs – per publication type and journal/venue, respectively. It can be observed that most models have been published in conference proceedings and in journals. It does do seem to be surprising because these are the most common publication types in

Table 1
Features of Main BN Topologies Used in Models for Effort Prediction

Name	*Description*	*Number of dependent variables*	*Contains causal relationships*	*Reflects system dynamics*
Converging Star (CS)	a star with the links directed from each predictor to a single dependent variable	1	no	no
Naïve Bayes (NB)	a structure of diverging star with the links directed from a single dependent variable to several predictors	1	no	no
Causal BN(CBN)	the structure is more complexis also called a general BN	multiple	yes	no
Dynamic BN(DBN)	a set of sequentially linked CBNs where each instance reflects the state of a system at specific point of time	multiple	yes	yes

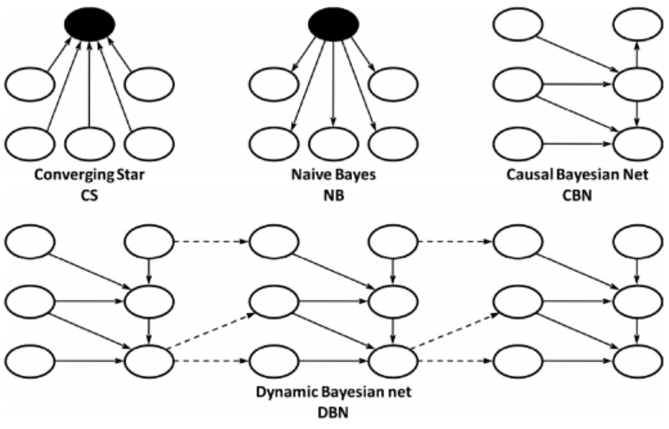


Figure 2: General Structures of Main BN Topologies used in Models for Effort Prediction

Source: based on [SMI RRP2BN]

almost all research fields. There is a dominating journal, IEEE Transactions on Software Engineering, which has published three papers on related models. However, these three papers account to only 13% of all papers analyzed in this study. Most journals and conference proceedings have published a single paper on related BNs. Thus, no particular journal or venue can be regarded as a central forum for presenting such models. It should be noted that the numbers in these figures do not reflect the number of developed models. In some papers several BNs have been discussed and some models have been discussed in multiple publications.

Figure 5 illustrates the number of related publications per year. From this figure we can clearly observe a growth in the number of published papers on related BNs. The majority of papers, i.e. over 65%, have been published in years 2008-2010.

Stewart [53] performed an analysis aimed to determine the accuracy of Naïve Bayes model for project delivery rate (productivity) prediction. The author developed 21 models for seven types of organizations in which the software was to be used. For each of these organization types three Naïve Bayes models have been built. The main difference between these three models was the number of predictor variables used – 64, 15, and seven, respectively. Although the main focus of the study was on BN models, two other techniques have also been used for comparison – model trees (which extend the concept of regression trees) and neural networks. All models have been generated and validated using the ISBSG dataset, Release 6 [24]. The author concludes that “Naive–Bayes classiûer is a valuable tool for analysis of software engineering data which can be used as an alternative to other more widely used approaches such as decision trees and neural networks”.

Stamelos *et al.* [52] have also analyzed the problem of software productivity prediction. However, they have used a very simple causal structure. More precisely, their model is a set of

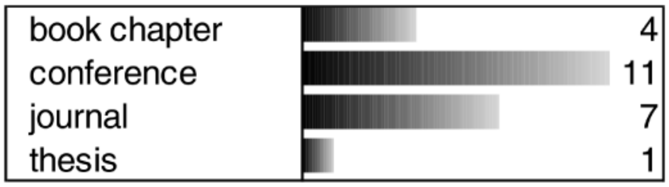


Figure 3: Number of Papers by Publication Type

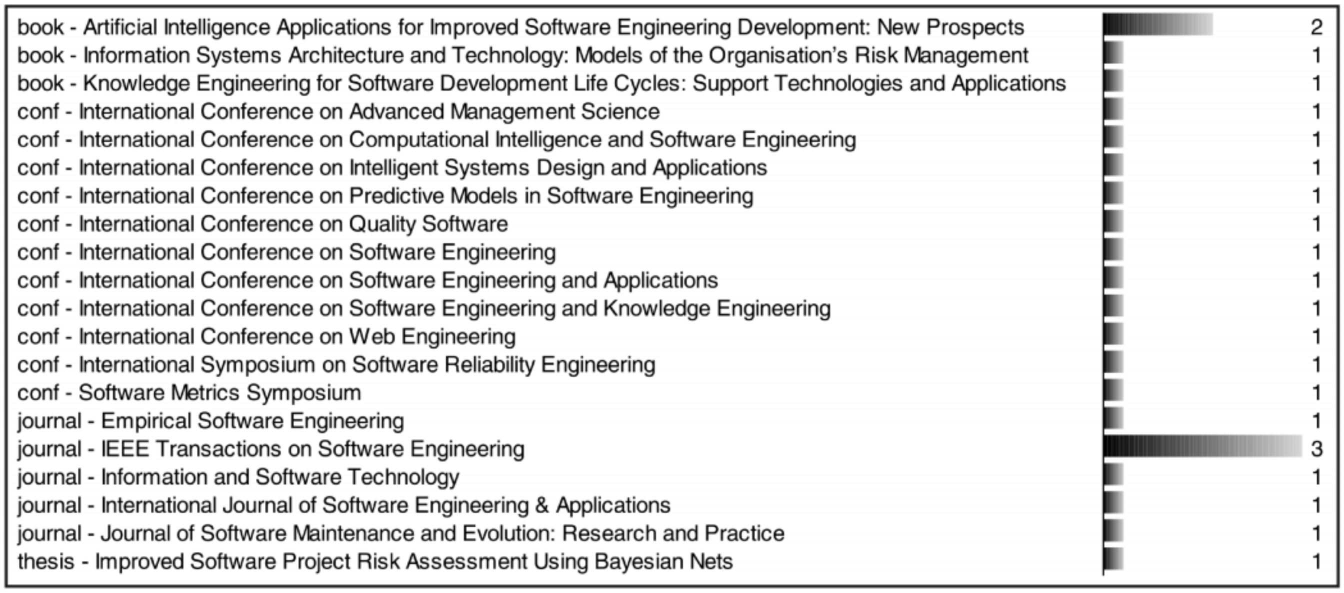


Figure 4: Published Papers by Journal / Venue

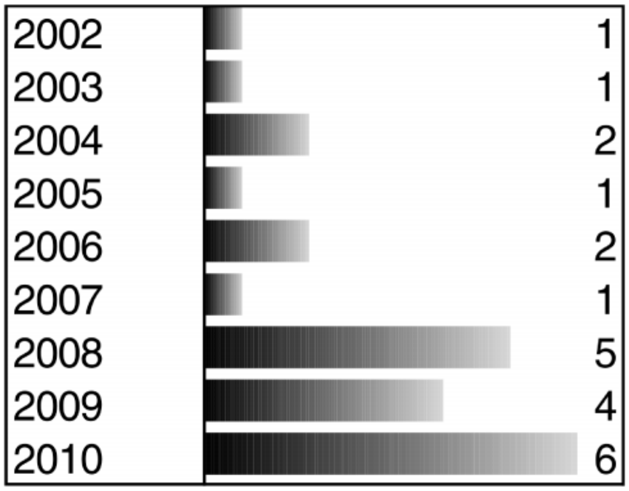


Figure 5: Published Papers by Year

three CS structures linked hierarchically. At the first (root) level there are "Product", "Computer", "Personnel", and "Project" factors – all expressed on a five-point ranked scale. The next level contains two variables "Technical factors" and "Human and Process Factors", also expressed on a ranked scale, which combine related pairs of variables from the first level. Finally, the "Productivity", defined over eight intervals, is a child node of two variables from the second layer. The authors performed model validation using publically available COCOMO81 dataset of software projects [16]. In an appendix the authors showed the structure of enhanced and more complex BN. However, this enhanced model was not investigated in detail in that study.

Bibi and Stamelos [5] built a BN purely for software development effort estimation. The model reflects the staged nature of software development process and uses knowledge on specific stage to predict subsequent stages. Author's approach has been illustrated with a model for Rational Unified Process (RUP).

RUP is an iterative process where nine disciplines are repeated in four phases of project lifecycle. Thus, the whole process can be naturally reflected as a DBN. The authors deliberately have not included two disciplines "configuration and change management" and "environment" because of their low impact on each iteration, and also to keep the model clear.

The model predicts the effort for each discipline in each iteration as well as total effort required to complete the project. As the time progresses, increasing number of observations entered to the model leads to more accurate predictions.

The model can be generalized by omitting the number of iterations. If only a single iteration is performed, the model reflects the waterfall lifecycle. This model can also be adjusted to reflect incremental or iterative lifecycle, where the number of iterations will depend on a particular project.

The authors have also presented more detailed version of this model where the most important actions in specific disciplines have been linked with variables reflecting the results of these actions. As a result, the model clearly illustrates the order of actions. The authors consciously put a stress on a proper model structure, keeping in mind that people would be the end users of the model, and thus the models should be as simple and clear as possible.

The MODIST Project-level model [15] has been developed to predict effort and quality in large software projects. Project managers are the target group of users. The model consists of six subnets: distributed communication and management, requirements and specification, process quality, people quality, functionality delivered, and quality delivered. The key focus of the model is on effort prediction. However, it offers significantly more analytical possibilities because it incorporates a trade-off component between quality delivered, number of employees, project duration and project functionality. Quality delivered is expressed as user satisfaction and software defectiveness (i.e. defect rate). Project functionality reflects project size expressed in the number of function points and other measures. The authors have discussed the rationale motivating definition of selected variables. They have also presented several examples illustrating how this model can be used for practical decision-making.

Pendharkar *et al.* [38] proposed a BN model with a structure of converging star to predict software development effort. The structure of this model is very simple – it contains a single dependent variable “Software Effort” and three predictors “Development Methodology”, “Type of CASE Tool”, and “CASE Tool Experience”. The authors compared the accuracy of predictions from this model with predictions from other models (neural network and regression tree). They conclude that “the point forecasts generated by the Bayesian model are competitive”.

Noothong and Sutivong [34] have adopted the MODIST Project-level model developed by Fenton *at al.* [15]. The authors have built a decision network by defining decision and utility nodes in the original model. They have discussed the main motivations for the extending nodes. In a set of hypothetical numerical examples they have discussed the benefits of model enhancement enabling improved decision support for project managers.

Wang *et al.* [55] developed a framework for software project level estimation. This framework consists of four BN sub-models that enable trade-off analysis between quality, effort, schedule and scope. These four sub-models, used as basic building blocks for constructing the project level model, are Component Estimation Model, Test Effectiveness Estimation Model, Residual Defect Estimation Model, and Test Estimation Model. The authors have demonstrated how their model could be used in practice to predict development effort as well as in trade-off analysis.

Mendes and Mosley [29] have developed eight BN models for predicting development effort in web projects. Four of these models have been generated fully automatically, i.e. both the structure and probability distributions, from two empirical datasets using Powersoft [10] and Hugin [23] tools. Four other models have been developed in a hybrid mode – structure defined by experts (the same in all these four models) and probability distributions learned from data also with Powersoft and Hugin tools. The results from these models have been compared with the results provided from other models/techniques manual stepwise regression, case-based reasoning, and mean and median-based effort models. Their study has been followed in [30] where the same accuracy has been achieved and in [28], where some models have been adjusted, but their accuracy has not been investigated.

Fenton *et al.* [16] proposed a model mainly built for defect prediction. In a normal use the model requires entering observations for software size and for process factors expressed on a 5-point ranked scale. These factors are grouped in the following subnets: scale of new functionality implemented, common influences (with project management factors), specification and documentation, design and development, testing and rework, and existing code base.

However, this model has also the ability for a limited effort prediction and the authors demonstrate this in a hypothetical scenario. The model enables effort prediction for four development activities: specification, coding, testing, and rework. Effort can be predicted only when user enters an observation for expected or tolerable number of residual defects.

The most important model limitation in respect with effort prediction is that effort is expressed on a ranked scale only from "very low" to "very high". It is interpreted as a degree of appropriateness of effort allocated for an activity in a project with specific inherent features. Thus, effort cannot be precisely expressed. As a result this model may not be as useful for effort prediction as for defect prediction.

Radliński *et al.* developed a BN [14, 44] that overcomes important limitations of the MODIST model [15] and defect prediction model [16]. It is not an enhanced version of the previous models but it has a completely new structure. It enables performing trade-off analysis with key variables, such as effort, project size, and software quality, expressed on a numeric scale. This model contains the following subnets: uncontrollable project factors, specification and documentation, coding, testing, and project trade-offs.

The distinctive feature of this model is the meaning of the project and process factors expressed on a ranked scale. Their scale is not absolute but relative. The value of such factor reflects the degree of dissimilarity of the currently analyzed project from the template project used as a base for assessments. Such interpretation may be less clear than using an absolute scale, but lower volume of data is required to customize and use the model.

This model can be customized to individual needs with the use of provided questionnaire and the ability to enter specific priors for productivity and defect rates. If a user cannot provide such priors the model can estimate them as probability distributions. It requires providing additional information on project features, such as language type, development platform, used methodology, and other. A sub-model with Naïve Bayes structure performs this estimation of priors [14, 42].

Fineman *et al.* [18] developed a basic model for trade-off analysis between resources (i.e. effort), process quality, software functionality (i.e. project size), and software quality. This model is only a conceptual version prepared to demonstrate how to properly reflect such trade-off in a BN. Each main variable is expressed on a ranked scale only what, together with limited number of incorporated detailed project factors, makes the model unusable in industrial setting. However, it can still be useful to explain basic laws related to project trade-offs and may serve as a basic skeleton for more complex models.

Hearty *et al.* [14, 20] developed a DBN model for predicting the velocity (productivity) in extreme programming (XP) projects. The model incorporates the empirical data available in the software engineering literature. However, both the structure and probability distributions have been defined by expert, not learnt from data. Model behavior has been illustrated and discussed in various scenarios. Model accuracy has been analyzed using empirical data from one real XP project (published in the paper). The authors have also discussed the motivation for adjusting the model after initial non-satisfactory results, the details on model adjustment, and the results from the final validation.

Abouelela and Benedicenti [1] have also built a DBN for XP projects. This model shares the target of earlier model by Hearty *et al.* [20]. However, it is structurally very different and contains more dependent variables. Apart from project velocity prediction, it also enables predicting software quality (as defected user stories) and the release time. The authors have performed model validation using empirical data from past two projects.

Torkar *et al.* [54] have developed yet another DBN for XP projects. However, the main aim of this model is different from previously discussed models that have been focused on project velocity. This model does not attempt to cover all project activities and enables prediction of testing effort. The authors discussed various scenarios illustrating model behavior. Model accuracy has been validated empirically using two industrial XP projects.

Schulz *et al.* [49] have proposed a Software Process Model with a main aim to predict defect correction effort. The model has a DBN structure, where each instance reflects a single development task related to implementing part of specific feature. The model predicts engineering effort, which is later used to predict defect correction effort, mainly based on the type of task. This type of task reflects task complexity. Interestingly, the model does not contain any numerical variable reflecting the size of project, feature or task. Model usage has been illustrated and discussed in a set of hypothetical scenarios. It has also been evaluated for accuracy using past project data.

In another study, Schulz *et al.* [48] have extended their research on predicting defect correction effort. They have developed a Defect Cost Flow Model with a CBN structure. This model is based on the assumption that the cost of correcting defects, which have been introduced early in the lifecycle, increases significantly in later stages of the project. The main relationships between variables have been discussed and illustrated numerically. Model behavior has been demonstrated using four hypothetical scenarios.

In contrast to his earlier studies, Radliñski has also analyzed how accurate BNs can be built with no input from domain expert [46]. Publically available empirical data on the past projects from a single company have been used [16]. The whole dataset has been randomly divided into training and testing samples. The training sample has been used to automatically generate full BN models, i.e. both the structure and parameters, using greedy thick thinning algorithm. The training sample has been used to test the accuracy of the generated models. The whole procedure has been repeated 10 times to ensure randomness of dataset split and for the analysis of repeatability of predictions.

The structures of these 10 generated models differ from each other, in some cases very significantly. This, in combination with low accuracy of predictions (discussed in the next section), suggest great differences among the projects in the dataset. Thus, even though the split of data into training and testing samples was random, many generated BNs have not learned generalized knowledge to accurately predict effort for projects in testing sample.

Wang *et al.* [56] have developed a CBN for predicting the variance of software project schedule. Both the structure and probability distributions have been automatically generated from past data using K2 search algorithm [11]. Expert knowledge has been partially incorporated through a set of required and forbidden links defined in structure learning stage. A very simple scenario illustrating model behavior has been discussed. In addition the authors have provided the results from the sensitivity analysis and brief evaluation of model accuracy of predictions.

5. COMPARISON OF MODELS

Table 2 provides a summary comparison of analyzed models by illustrating model topology, source for defining topology and parameters and an indication whether a model has been validated empirically. Topology acronyms have been explained earlier in Section 3. The topology for most models have been defined by an expert (or experts). Mendes and Mosley [29] built four models – two with topology built by experts and another two with topology learnt from data. For models with NBC topology this involved only selecting the predictor variables that will be used in the model. In models with more complex topologies this step also required defining links between appropriate variables. When a topology is learnt from data an expert can still adjust model structure either before the learning process or after it. For example, Wang et al. [56] have learned the topology of their model form data, but prior to that, they have specified that some variables must be connected and some other must not. Then these requirements have been used by a learning algorithm.

When defining model parameters, i.e. probability distributions, most authors encoded either their own knowledge and experience or from external experts. In many cases these beliefs have been supported by empirical data (denoted as "expert/data"). For many models probability distributions have been learned from empirical data using parameter learning algorithms. However, in the NBC model by Radliñski [46] the distributions mainly come from empirical data but they have been adjusted by an expert to accumulate both expert knowledge and results from other analyses.

Many models have not been validated for accuracy against empirical data. Either only an early-stage conceptual version of a model has been published or for larger and more mature models no empirical data was available for validation. Thus, these two groups of models have been validated internally by discussing hypothetical scenarios and predictions for these scenarios. All models with any sort of validation based on empirical data have been marked with "yes" in the last column of Table 2. The predictive accuracy of these models have been presented later in Table 4.

Table 2
Summary Comparison of Analyzed Models

Authors	*Purpose*	*Topology*	*Source for topology*	*Source for parameters*	*Validated empirically*
Abouelela and Benedicenti [1]	productivity, duration, quality	DBN	expert	expert	yes
Bibi and Stamelos [5]	effort	DBN	expert	unknown	no
Fenton *et al.* [15]	effort, duration, quality	CBN	expert	expert/data	no
Fenton *et al.* [16]	number of defects, partially: effort	CBN	expert	expert/data	no*
Fineman *et al.* [18]	effort, quality	CBN	expert	expert	no
Hearty *et al.* [14, 20]	productivity	DBN	expert	expert/data	yes
Mendes and Mosley [29]	effort	CBN	expert & data	data	yes
Mendes [28]	effort	CBN	expert & data	data	no
Mendes [30]	effort	CBN	expert & data	data	yes
Noothong and Sutivong [34]	effort	DN	expert	expert/data	no
Pendharkar *et al.* [38]	effort	CS	expert	data	yes
Radliński *et al.* [14, 42, 44]	productivity, defect rate	NBC	expert	data/expert	no
Radliński *et al.* [14, 43, 44]	effort, quality	CBN	expert	expert/data	no
Radliński [46]	effort, quality	CBN	data	data/expert	yes
Schulz *et al.* [48]	effort	CBN	expert	expert/data	no
Schulz *et al.* [49]	effort	DBN	expert	expert/data	yes
Stamelos *et al.* [52]	productivity	CBN	expert	expert/data	yes
Stewart [53]	productivity	NBC	expert	data	yes
Torkar *et al.* [54]	effort	DBN	expert	expert/data	yes
Wang *et al.* [55]	effort, quality	CBN	expert	expert/data	no
Wang *et al.* [56]	variance of project schedule	CBN	data/expert	data	yes

* The authors performed validation using empirical data only for predicted number of defects, what is beyond the scope of this paper.

Table 3 illustrates how different groups of factors describing software projects have been used in specific models. It can be observed that most models incorporate variables describing various aspects of projects. Project size is captured using various measures, such as function points, lines of code, user stories, requirements and other. Process factors refer to the organization of the development process and its maturity. People factors are typically focused on the level of staff experience and motivation. Project factors refer to complexity, stability, tools used, or project type. In 11 studies (55%) developed models contain variables from all fours groups. Surprisingly, not all models for effort prediction use project size as a main predictor. For example, Schulz *et al.* [48, 49] built two separate models and none of them contains a variable reflecting project size.

Most authors published either a full list of variables or illustrated a model structure from which all variables can be read. Stewart [53] performed three experiments involving building models with 64, 15 and seven input variables, respectively. Complete list of variables has been published only for seven-variable models.

Table 3
Groups of Factors used in Models

Authors	*Project size*	*Process factors*	*People factors*	*Project factors*
Abouelela and Benedicenti [1]	+	+	+	-
Bibi and Stamelos [5]	+	+	-	+
Fenton *et al.* [15]	+	+	+	+
Fenton *et al.* [16]	+	+	+	+
Fineman *et al.* [18]	+	+	+	-
Hearty *et al.* [14, 20]	+	+	+	-
Mendes and Mosley [29]	+	+	+	+
Mendes [28]	+	+	+	+
Mendes [30]	+	+	+	+
Noothong and Sutivong [34]	+	+	+	+
Pendharkar *et al.* [38]	–	+	+	+
Radlinski *et al.* [14, 42, 44]	+	-	-	+
Radlinski *et al.* [14, 43, 44]	+	+	+	+
Radlinski [46]	+	+	+	+
Schulz *et al.* [48]	-	+	-	-
Schulz *et al.* [49]	-	+	-	+
Stamelos *et al.* [52]	+	+	+	+
Stewart [53]	?	?	?	+
Torkar *et al.* [54]	–	+	+	+
Wang *et al.* [55]	+	+	+	+
Wang *et al.* [56]	+	+	+	+

As shown in Table 2, only some BNs have been validated empirically. Table 4 provides the results of validation of those BNs for which it has been performed. Most common measures used to validate these models are:

- mean magnitude of relative error (MMRE),
- median magnitude of relative error (MdMRE),
- prediction at level *l* (Pred(l)),
- accuracy.

More accurate predictions are provided by models with lower MMRE and MdMRE, and with higher Pred(l) and accuracy.

It can be observed that the values of these measures vary depending on the study. Even a single very accurate model never provides accurate predictions for all analyzed cases. For example a model by Hearty *et al.* [20] provides almost perfect prediction after learning with two development iterations but this accuracy decreases with using data from further iterations to learn the model. Similarly, a model by Schulz et al. [49] enables very high accuracy of predictions for features 1 and 2, but significantly worse for features 3–5.

On the other side, these is a set of models, especially developed by Mendes and Mosley [29] and Mendes [30], with generally very low accuracy. Such low accuracy makes them almost unusable in industrial use. However, the authors performed a comparison with the results generated from other techniques, and it appeared that other techniques also provided poor predictions. Depending on the dataset used and the accuracy measure, some BNs were more accurate than other techniques. However, stepwise regression model produced the most accurate predictions of all techniques investigated in these studies. Additionally, use of simple models, such as the median effort, can outperform more complex models, such as BNs, depending on dataset and accuracy measure used. In another study involving comparison of accuracy of various techniques, BN models outperformed other techniques in the studies by Stewart [53] and Pendharkar *et al.* [38].

Not all authors who performed the empirical validation provided detailed evaluation results. For example, Hearty et al. [20] provided numeric values for MMRE only for selected scenarios, and presented other scenarios in graphical form. Thus, the values of MMRE published in this study have been cited exactly only for model trained with data from two iterations and with no training. All other values have been read from the graph published in original paper.

Some models have been validated with empirical data but only for very few number of projects. For these studies calculated values of accuracy measures would be meaningless. Thus, for them we provide the actual and predicted values.

Table 4
Results of Empirical Validation of BNs

Authors	*Selected validation results*				
Abouelela and Benedicenti [1]	System 1		number of days:	actual=60	predicted=65
	System 2		number of days:	actual=12	predicted=11
Hearty *et al.* [20]	without learning			MMRE=0.51	
	learning with 1 iteration			MMRE≈0.11	
	learning with 2 iterations			MMRE=0.026	
	learning with 3 iterations			MMRE≈0.19	
	learning with 4 iterations			MMRE≈0.11	
	learning with 5 iterations			MMRE≈0.035	
	learning with 6 iterations			MMRE≈0.65	
	learning with 7 iterations			MMRE≈0.25	
Mendes and Mosley [29]; Mendes [30] – only for BNAuHu and BNHyHu models	Set 1	BNAuPo	MMRE=13.97	MdMRE=2.57	Pred(25)=4.62%
		BNAuHu	MMRE=7.65	MdMRE=1.67	Pred(25)=7.69%
		BNHyPo	MMRE=36.00	MdMRE=4.90	Pred(25)=7.69%
		BNHyHu	MMRE=1.90	MdMRE=0.86	Pred(25)=15.38%
	Set 2	BNAuPo	MMRE=14.93	MdMRE=6.46	Pred(25)=0%
		BNAuHu	MMRE=4.09	MdMRE=0.96	Pred(25)=1.54%
		BNHyPo	MMRE=37.31	MdMRE=8.05	Pred(25)=1.54%
		BNHyHu	MMRE=27.95	MdMRE=5.31	Pred(25)=3.08%
Pendharkar *et al.* [38]			MMRE=1.56		
			accuracy=86% (of predicting actual effort interval)		
Radliński [46]	effort		MMRE=0.97	MdMRE=0.60	Pred(25)=17%
	productivity		MMRE=0.77	MdMRE=0.40	Pred(25)=27%
Schulz *et al.* [49]	feature 1	accuracy(engineering effort)=95%	accuracy(correction effort)=97%		
	feature 2	accuracy(engineering effort)=100%	accuracy(correction effort)=92%		
	feature 3	accuracy(engineering effort)=88%	accuracy(correction effort)=77%		
	feature 4	accuracy(engineering effort)=85%	accuracy(correction effort)=84%		
	feature 5	accuracy(engineering effort)=63%	accuracy(correction effort)=14%		
Stamelos *et al.* [52]		52%<accuracy<67%			
Stewart [53]	Banking		MMRE (Model_1)=0.52	MMRE(Model_2)=0.41	
			MMRE(Model_3)=0.42		
	Communication		MMRE (Model_1)=0.52	MMRE(Model_2)=0.41	
			MMRE(Model_3)=0.42		
	Electricity/Gas/Water		MMRE (Model_1)=0.52	MMRE(Model_2)=0.41	
			MMRE(Model_3)=0.42		
	Financial/property/business		MMRE (Model_1)=0.52	MMRE(Model_2)=0.41	
			MMRE(Model_3)=0.42		
	Insurance		MMRE (Model_1)=0.52	MMRE(Model_2)=0.41	
			MMRE(Model_3)=0.42		
	Manufacturing		MMRE (Model_1)=0.52	MMRE(Model_2)=0.41	
			MMRE(Model_3)=0.42		
	Public administration		MMRE (Model_1)=0.52	MMRE(Model_2)=0.41	
			MMRE(Model_3)=0.42		
Torkar *et al.* [54]	Project A		MMRE=0.073		
	Project B		MMRE=0.2996		
Wang *et al.* [56]	Project A	schedule variance:	actual=7.4%	predicted="<10%"	
	Project B	schedule variance:	actual=12.5%	predicted="10%-20%"	

Each analyzed BN has been developed for a special purpose, under specific constraints and targets specific environment. Hence, none of these models covers all possible problems related to software effort prediction. In other words, none of these models is universal in its purpose, structure and intended way of use. Thus, potential users either have to thoroughly analyze which BN suits their needs or to develop a custom model themselves.

6. LIMITATIONS OF BUILDING AND USING BNS

The main advantage of using BNs seems to be related with the ability of incorporating expert knowledge and combining it with empirical data. Incorporation of expert knowledge is time consuming, both when building model structure and when defining probability distributions. The need of defining only necessary and direct relationships in complex models is difficult, especially without support from experienced expert in artificial intelligence, machine learning or statistics. This is related to the fact that in properly defined structures the real-world relationships should be reflected in the model, either with or without a direct link between related variables. Similarly, variables unrelated in reality should also be unrelated in the model, i.e. observation entered to one of them should not be passed to the second of them, neither directly nor through a set of intermediate nodes.

A difficulty in defining probability distributions by an expert is related with the high number of probability values that need to be entered for variables having high number of parents and high number of states. For example, a binary variable with two parents contains a probability table with eight entries (2×2×2). Adding another binary parent increases the size of probability table to 16 (2×2×2×2). If all variables have three possible states the size of probability tables for this variable would increase to 27 (3×3×3) and 81 (3×3×3×3) entries, respectively. Defining tables of such size manually by an expert is not only time consuming but also prone to inconsistencies.

These problems have been identified several years ago. Various researches attempt to simplify the process of building BNs by experts. These efforts are focused on support in defining just the model structure [33, 41, 45], just the probability distributions [13, 17, 21, 35, 39, 57], or both structure and probabilities [26, 31, 51].

Such techniques may have a great potential for simplifying the process and reducing the time of BN development by expert. Popular machine learning and statistical methods enable faster model development from data. Longer time required to build a BN model by expert may potentially result in developing a model that better reflects analyzed problem than models automatically generated from data. However, there is an inevitable trade-off between model functionality and the time required to build it. Project managers are faced with the question, if they can wait longer for a more functional model or if they need a less functional model but which can be built fast to address the current decisional problem.

Although BN community is growing, BNs are still significantly less popular as other modeling techniques. Partially, this is related with the modeling difficulties discussed above. But solving these issues will only solve one side of the whole problem. Another side is related to using the models for decision support. Bayesian nets generate the output in the form of probability distributions. People are typically used to obtaining results as point values. For example, a manager often wishes to know how much effort is required to develop a particular project. Most modeling techniques would provide an answer as a specific point value, say 1 000 person-hours. However, BNs provide predictions in more sophisticated form of probability distributions. In this example, a model might produce a following result: 100 500 person-hours with probability 10%, 500-2000 person-hours with probability of 70%, and 2000-4000 person-hours with probability of 20%. Such result provides more information than just a point value because it also provides the values of probability for each interval. Nevertheless, the interpretation of such result is more difficult because a manager receives more numerical values for analysis. Additionally, most models analyzed in this paper have important numeric variables discretized into sets of very few fixed intervals. This reduces the precision of expressing such values because in a wide interval all values are treated as equal. For example in an interval 500-2000 values 550 and 1900 are treated as equal. Techniques that do not require such discretization of numeric variables have the advantage in this matter. Thus, there is a need to related to the ability of presenting results of predictions from BNs also in simpler ways.

Winkler [58] has also noted inadequate tool support for building BNs. While his analysis was related to BN applications in healthcare, this and most other general problems apply also to software engineering and project management. Since then significant effort has been put to build commercial scale and quality BN tools that not just enable performing calculations or learning from data but are focused on easier modeling and performing simulations by expert [2, 4, 23]. While some of these tools offer great functionality, none of them solves all problems related to the required high level of usability.

7. SUMMARY

In this paper we have shown the BN models that have been used for software development effort prediction. Most of these models are very different from each other because they have been built under different constraints and for different environments. As a result they incorporate various types of structures and sources of data.

About a half of analyzed models have been empirically validated by authors in terms of predictive accuracy. In most of these cases, the authors seem to be satisfied with the accuracy of their models. Yet, some models are very inaccurate as indicated by commonly used measures of predictive accuracy. However, in comparison with other modeling techniques even these models have shown to be competitive.

What is very useful about the BNs is the ability to express expert knowledge in a formal predictive model using a rigorous probability calculus. This is especially worthy is situations where no empirical data from past projects is available, and thus no data-driven machine learning techniques cannot be applied. However, the process of elicitation and formalization of expert knowledge is not trivial. This needs to be simplified by developing efficient techniques that solve this problem. These new techniques need to be available in commercial scale and quality BN tools. Fortunately, both researchers and commercial vendors of such BN tools have identified this problem and focus their effort on providing useful solutions. Still, some work needs to be done to allow building and using BNs by managers without background in artificial intelligence and statistics.

References

[1] Abouelela M., Benedicenti L., Bayesian Network Based XP Process Modelling, *International Journal of Software Engineering & Applications*, 1(3): 1-15, 2010.

[2] AgenaRisk, Bayesian Network and Simulation Software for Risk Analysis and Decision Support, Agena, 2010, *http://www.agenarisk.com.*

[3] Bayes T., An Essay Towards Solving a Problem in the Doctrine of Chances. By the late Rev. Mr. Bayes, F.R.S. communicated by Mr. Price, in a letter to John Canton, A. M. F. R. S., Phil. Trans. Royal Soc. London, 53: 370-418, 1763.

[4] BayesiaLab, Bayesia SAS, 2010, *http://bayesia.com.*

[5] Bibi S., Stamelos I., Software Process Modeling with Bayesian Belief Networks, Proc. Int. Software Metrics Symposium, Chicago, 2004.

[6] Boehm B., Software Engineering Economics, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[7] Ceylan E., Kutlubay F. O., Bener A.B., Software Defect Identification Using Machine Learning Techniques, Proc. 32nd EUROMICRO Conference on Software Engineering and Advanced Applications, Crotia, Aug. 2006, 240–247.

[8] Challagulla V. U. B., Bastani F. B., I-Ling Yen, Paul R. A., Empirical Assessment of Machine Learning based Software Defect Prediction Techniques, Proc. 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, Feb. 2005, 263 270.

[9] Cheng J., Grainer R., Comparing Bayesian Network Classifiers, Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Morgan Kaufmann, 1999, 101-108.

[10] Cheng J., Powersoft Bayesian Belief Network Software, 2001. *http://www.cs.ualberta.ca/~jcheng/bnsoft.htm.*

[11] Cooper G., Herskovits E., A Bayesian Method for the Induction of Probabilistic Networks from Data, *Machine Learning*, 4, 1992, 309-347.

[12] Darwiche A., Modeling and Reasoning with Bayesian Networks. Cambridge University Press, 2009.

[13] Das B., Generating Conditional Probabilities for Bayesian Networks: Easing the Knowledge Acquisition Problem, 2004, *http://www.citebase.org/cgi bin/citations?id= oai:arXiv.org:cs/0411034.*

[14] 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.

[15] Fenton N., Marsh W., Neil M., Cates P., Forey S., Tailor M., Making Resource Decisions for Software Projects, Proc. 26th Int. Conference on Software Engineering, Washington, DC, *IEEE Computer Society*, 397–406, 2004.

[16] Fenton N., Neil M., Marsh W., Hearty P., Radliński Ł., Krause P., On the Effectiveness of Early Life Cycle Defect Prediction with Bayesian Nets, Empir. *Software Eng.*, 13: 499–537, 2008.

[17] Fenton N. E., Neil M., Caballero J. G., Using Ranked Nodes to Model Qualitative Judgments in Bayesian Networks, *IEEE Transactions on Knowledge and Data Engineering*, 19(10), 1420-1432, 2007.

[18] Fineman M., Fenton N., Radliński Ł., Modelling Project Trade-off Using Bayesian Nets, Proc. International Conference on Computational Intelligence and Software Engineering, IEEE, Wuhan, China, 2009: 1–4.

[19] Friedman N., Geiger D., Goldszmidt M., Bayesian Network Classifiers, *Machine Learning*, 29, 1997: 131-163.

[20] Hearty P., Fenton N., Marquez D., Neil M., Predicting Project Velocity in XP using a Learning Dynamic Bayesian Network Model, *IEEE Trans. Software Eng.*, 37(1): 124–137, 2009.

[21] Helsper E. M., van der Gaag L. C., Feelders A. J., Loeffen W. L. A., Geenen P. L., Albers A. R. W., Bringing Order into Bayesian-Network Construction, Proc. 3rd Int. Conf. on Knowledge Capture, Banff, Alberta, Canada, 2005: 121-128.

[22] Hewett R., Kijsanayothin P., On Modeling Software Defect Repair Time, *Empirical Software Engineering*, 14(2), 2009: 165-186.

[23] Hugin, HuginExpert A/S, 2010, *http://www.hugin.com.*

[24] International Software Benchmarking Standards Group, 2002. *http://www.isbsg.org.au.*

[25] Jensen F. V., An Introduction to Bayesian Networks, UCL Press, London, 1996.

[26] Kraaijeveld P., Druzdzel M., Onisko A., Wasyluk H., GeNIeRate: An Interactive Generator of Diagnostic Bayesian Network Models, Working Notes of the 16th International Workshop on Principles of Diagnosis (DX-05), Monterey, CA, 2005: 175-180.

[27] Mair C., Kadoda G., Lefley M., Phalp K., Schofield C., Shepperd M., Webster S., An Investigation of Machine Learning based Prediction Systems, *Journal of Systems and Software*, 53(1), 2000, 23–29.

[28] Mendes E. Using Bayesian Networks for Web Effort Estimation, in: Meziane F., Vadera S. (Eds.), Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects, Information Science Reference, New York, 2008, 26-44.

[29] Mendes E., Mosley N., Bayesian Network Models for Web Effort Prediction: A Comparative Study, *IEEE Trans. Software Eng.*, 34, 2008, 723-737.

[30] Mendes E., The Use of Bayesian Networks for Web Effort Estimation: Further Investigation, Proc. 8th Int. Conf on Web Engineering, Yorktown Heights, NJ, pp. 203-216, 2008.

[31] Nadkarni S., Shenoy P. P., A Causal Mapping Approach to Constructing Bayesian Networks, Decision Support Systems, 38(2), 259 281, 2004.

[32] Neapolitan R. E., Learning Bayesian Networks, Pearson Prentice Hall, Upper Saddle River, 2004.

[33] Neil M., Fenton N., Nielsen L., Building Large-Scale Bayesian Net-works, *Knowledge Engineering Review*, 15(3), 257-284. 2000.

[34] Noothong T., Sutivong D., Software Project Management Using Decision Networks, Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications, Washington, DC: IEEE Computer Society, 2006, 1124-1129.

[35] O' Hagan A., Buck C. E., Daneshkhah A., Eiser J. E., Garthwaite P. H., Jenkinson D. J., Oakley J. E., Rakow T., Uncertain Judgements: Eliciting Expert Probabilities, Chichester: John Wiley and Sons, 2006.

[36] Pearl J., Bayesian Networks: A Model of Self-Activated Memory for Evidential Reasoning. (UCLA Technical Report CSD-850017). Proc. 7th Conf. of the Cognitive Science Society, University of California, Irvine, pp. 329–334, 1985.

[37] Pearl J., Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco, 1988.

[38] Pendharkar P. C., Subramanian G. H., and Rodger J.A., A Probabilistic Model for Predicting Software Development Effort, *IEEE Transactions on Software Engineering*, 31(7), 2005, 615-624.

[39] Pfautz J., Cox Z., Catto G., Koelle D., Campolongo J., Roth E., User-Centered Methods for Rapid Creation and Validation of Bayesian Belief Networks, Proc. 5th Bayesian Modeling Applications Workshop, 23rd Annual Conference on Uncertainty in Artificial Intelligence, Vancouver, British Columbia, 2007.

[40] Radliński Ł., A Survey of Bayesian Networks for Risk Assessment in Software Engineering, Research Journals of the University of Szczecin. Studia Informatica, 21, 2009: 119-129 (in Polish).

[41] Radliński Ł., Fenton N., Causal Risk Framework for Software Projects, in Wilimowska Z., Borzemski L., Grzech A., Świątek J. (Eds.), Information Systems Architecture and Technology. IT Technologies in Knowledge Oriented Management Process, Wrocław, Poland: Oficyna Wydawnicza Politechniki Wrocławskiej, 2009, 49-59.

[42] Radliński Ł., Fenton N., Marquez D., Estimating Productivity and Defect Rates Based on Environmental Factors, in Information Systems Architecture and Technology: Models of the Organisation's Risk Management, Wrocław, Poland: Oficyna Wydawnicza Politechniki Wrocławskiej, 2008: 103-113.

[43] Radliński Ł., Fenton, N., Neil, M., Marquez D., Improved Decision-Making for Software Managers Using Bayesian

Networks, Proc. 11th IASTED Int. Conf. Software Engineering and Applications, Cambridge, MA, 13–19, 2007.

[44] Radlinski L., Improved Software Project Risk Assessment Using Bayesian Nets, Ph.D. Thesis, Queen Mary, University of London, London, 2008.

[45] Radliński Ł., RRP2BN – A Method for Generating Bayesian Nets from Risk Response Plan, Project Management. Selected Issues, Szyjewski Z., Swacha J. eds., Szczecin: Zapol, 2010: 133-154.

[46] Radliński Ł., Software Development Effort and Quality Prediction Using Bayesian Nets and Small Local Qualitative Data, Proc. International Conference on Software Engineering and Knowledge Engineering, Redwood City, CA, 2010: 113-116.

[47] Russell S., Norvig P., Artificial Intelligence. A Modern Approach, Second Edition, Pearson Education, Inc., Upper Saddle River, 2003.

[48] Schulz T., Radliński Ł., Gorges T., Rosenstiel W., Defect Cost Flow Model – A Bayesian Network for Predicting Defect Correction Effort, Proc. 6th International Conference on Predictive Models in Software Engineering, Timisoara, 2010.

[49] Schulz T., Radliński Ł., Gorges T., Rosenstiel W., Software Process Model using Dynamic Bayesian Networks, [in] Ramachandran M. (ed.), Knowledge Engineering for Software Development Life Cycles: Support Technologies and Applications, IGI-Global, 2010 (accepted).

[50] Shepperd M., Kadoda G., Comparing Software Prediction Techniques Using Simulation, *IEEE Transactions on Software Engineering*, 27(11), 2001: 1014 1022.

[51] Skaanning C., A Knowledge Acquisition Tool for Bayesian-Network Troubleshooters, Proc. 16th Conference on Uncertainty in Artificial Intelligence. Stanford, CA: Stanford University, 2000.

[52] Stamelos I., Angelis L., Dimou P., Sakellaris E., On the use of Bayesian Belief Networks for the Prediction of Software Productivity, *Information and Software Technology*, 45(1): 51-60, 2003.

[53] Stewart, B., Predicting Project Delivery Rates using the Naive–Bayes Classifier, *J. Software Maint. Evol.: Res. Pract.*, 14: 161–179, 2002.

[54] Torkar R., Adnan N. M., Alvi A. K., Afzal W., Predicting Software Test Effort in Iterative Development using a Dynamic Bayesian Network, Proc. of 21st *IEEE International Symposium on Software Reliability Engineering*. Industry Practice Track, San Jose, 2010.

[55] Wang H., Peng F., Zhang C., Pietschker A, Software Project Level Estimation Model Framework based on Bayesian Belief Networks, Proceedings of the Sixth International Conference on Quality Software (QSIC '06), IEEE Computer Society, Washington, DC, USA, 2006: 209-218.

[56] Wang X., Wu C., Ma L., Software Project Schedule Variance Prediction using Bayesian Network, Proc. 2010 *IEEE International Conference on Advanced Management Science* (ICAMS), Chengdu, 2010: 26-30.

[57] Wiegmann D.A., Developing a Methodology for Eliciting Subjective Probability Estimates During Expert Evaluations of Safety Interventions: Application for Bayesian Belief Networks, (Rep. No. AHFD-05-13/NASA-05-4). Aviation Human Factors Division Institute of Aviation, University of Illinois, 2005.

[58] Winkler R., Why Bayesian Analysis Hasn't Caught on in Healthcare Decision Making, *International Journal of Technology Assessment in Health Care*, 17(1): 56 66, 2001.

[59] Zhang D., Tsai J. J. P., Machine Learning and Software Engineering, *Software Quality Journal*, 2003, 11(2): 87-119.