The Journal of Systems and Software 127 (2017) 109–119

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Bayesian network model for task effort estimation in agile software development

Srdjana Dragicevic[a], Stipe Celar[b,*], Mili Turic[c]

[a] Split Airport, Cesta dr. Franje Tudmana 1270, 21217 Kastel Stafilic, Croatia
[b] Department of Electronics, FESB, University of Split, R. Boskovica 32, 21000 Split, Croatia
[c] Venio indicium d.o.o., Doverska 19, 21000 Split, Croatia

ARTICLE INFO

Article history:
Received 11 December 2015
Revised 24 January 2017
Accepted 30 January 2017
Available online 31 January 2017

Keywords:
Bayesian network
Effort prediction
Agile software development

ABSTRACT

Even though the use of agile methods in software development is increasing, the problem of effort estimation remains quite a challenge, mostly due to the lack of many standard metrics to be used for effort prediction in plan-driven software development. The Bayesian network model presented in this paper is suitable for effort prediction in any agile method. Simple and small, with inputs that can be easily gathered, the suggested model has no practical impact on agility. This model can be used as early as possible, during the planning stage. The structure of the proposed model is defined by the authors, while the parameter estimation is automatically learned from a dataset. The data are elicited from completed agile projects of a single software company. This paper describes various statistics used to assess the precision of the model: mean magnitude of relative error, prediction at level *m*, accuracy (the percentage of successfully predicted instances over the total number of instances), mean absolute error, root mean squared error, relative absolute error and root relative squared error. The obtained results indicate very good prediction accuracy.

1. Introduction

In the recent years, agile methodologies have been widely accepted in software development. According to a survey (Version One, 2007), only 3.4% of the surveyed companies have never used the agile methodology in software development projects.

The term "agile methods" refers to a number of methods that share the same goals and values (Beck et al., 2001). They all are based on development iterations, incremental improvements and continuous feedback and share a lack of formal documentation and specification. Requirements are elicited and specified at the beginning of each iteration. That ensures quick responses to requirements changes and minimal waste of time, but results in (Helmy et al., 2012; Nawrocki et al., 2002; Sillitti and Succi, 2005):

- A lack of a bigger picture; connections between requirements may be missing.
- Implicit elicitation of non-functional requirements; eliciting and managing techniques are not provided.
- Problems in product maintenance; documentation is limited.
- Lack of engineering activities that help create a specification of the expected requirements.

In addition, agile development shares the same problems regarding communications between stakeholders with plan-driven software development, because it uses the same elicitation techniques (Sillitti and Succi, 2005).

To resolve the above mentioned problems, we have created the Method for Elicitation, Documentation and Validation of Software User Requirements (MEDoV) (Dragicevic et. al., 2011; Dragicevic and Celar, 2013). MEDoV was successfully applied in an agile software development project (Dragicevic et al., 2014). The project was completed on time, within the estimated budget, and no unnecessary features were developed. Of course, the success of the application of the method in a single project is not sufficient for its final validation. Before applying the method to other projects, it is necessary to define metrics by which the efficiency of the method will be measured in an agile environment.

Incorrect and incomplete requirements are the major causes of failure of software projects (Standish Group, 2009). Therefore, a good method for eliciting and documenting user requirements will certainly contribute to the success of the entire project. Moreover, in agile software development the requirements prioritization is the client's responsibility (Sillitti and Succi, 2005). These are the reasons why this metric should be able to measure the project

* Corresponding author.
E-mail addresses: srdjana.dragicevic@split-airport.hr (S. Dragicevic), stipe.celar@fesb.hr (S. Celar), mili.turic@venio.hr (M. Turic).

http://dx.doi.org/10.1016/j.jss.2017.01.027
0164-1212/

success. The success of a software project depends on project effort, cost and the quality of the final product. When the project is completed, it is easy to determine its success. What we would like to confirm is that before the final validation of a project success, one can use this metric to predict the success.

Traditional software project prediction models are proven either to be unreliable, or require sophisticated metrics to be rendered reliable (Borade and Khalkar, 2013), both representing a problem in agile development. Many metrics used in traditional software development project planning simply cannot be used in agile development project planning.

Agile development teams usually use story points to measure the effort needed to implement a user story. Story points are useful to compare technical complexity, effort and uncertainty of different user stories, as well as to measure project velocity. Sometimes, project managers assign x hours for y story points and estimate the hours required to complete the task. But this technique is not appropriate to define effort in hours or days absolutely, because story points correspond to time distribution and equivalence, e.g., 1 story point = 5 h is not valid for all times, all teams and all projects. Instead of that, a project manager can use velocity to estimate how many story points can be implemented in a sprint. But velocity needs three or four sprints to stabilize, and it is not suitable for use at the beginning of a project.

Consequently, agile teams use a technique called ideal days/hours (Cohn, 2005). The developers estimate how many days/hours are required to finish the task if they are focused exclusively on that task, without interruptions such as by meetings, drop-ins, phone calls, etc. The task that requires eight ideal hours can take two or three days because of interruptions. It can take even more days if a developer works on several tasks at the same time, not to mention that developers usually underestimate the needed effort.

Moreover, Jorgensen (2013) shows that effort estimation depends on the direction of comparison. When developers compare story A to story B, the effort estimation is different from that when they compare story B to story A. The results are particularly inaccurate when comparing a large user story with a much smaller one.

Therefore, the main objective of this research is to find a technique that will facilitate the assessment of the required effort. This technique should be suitable for use even in the planning stage, and help project managers in further agile software development. It should not affect the agility and it should be suitable for any agile method.

Typical problems of traditional effort, cost and quality prediction models can be overcome by using the BN models (Fenton and Neil, 1999; Fenton et al., 2008) due to the:

- Flexibility of the BN building process (based purely on expert judgment, empirical data, or the combination of both).
- Ability to reflect causal relationships.
- Explicit incorporation of uncertainty as a probability distribution for each variable.
- Graphical representation that makes the model clear.
- Ability of both, forward and backward inferences.
- Ability to run the model with missing data.

It has been shown (Celar et al., 2012; Jorgensen, 2010,2014) that relevant empirical data can significantly increase the accuracy of predictions. Consequently, data from real agile projects are used for building this BN model. The model is intended for the prediction of smaller parts of projects (project tasks) and not for their scheduling. For that reason, the terms “effort” and “duration” are used interchangeably in this paper.

Various statistics are used to determine the accuracy of prediction in software estimation. The most commonly used metrics are: the Magnitude of Relative Error (MRE), the Mean Magnitude of Relative Error (MMRE), and the Prediction at Level m (Pred. (m)), although some authors suggest that other statistics represent more appropriate metrics (Foss et al., 2003; Kitchenham et al., 2001; Korte and Port, 2008).

The remainder of this paper is structured as follows: the investigation of current usages of Bayesian network (BN) models for software effort prediction is described in Section 2, BN is explained in Section 3, conditions which the proposed BN model should meet are given in Section 4, and the building process is described in detail in Section 5. The conclusions as well as the outlines of future work are presented in Section 6.

2. Related works

Effort estimation is inherently inaccurate. There are many reasons that cause imprecision: the lack of relevant information, some metrics are of subjective nature, the complex interaction between metrics and the amount of effort required to gather metrics. Bayesian network has been proposed to reduce these uncertainties, since the BN automatically deals with the uncertainty and risk due to its own statistical nature.

Mendes et al. (2012) depict the successful use of BN models for web development effort estimation and mention previous papers which describe how hybrid Bayesian network models (structure expert-driven and probabilities data-driven) have outperformed the mean and median-based effort, multivariate regression, case-based reasoning, and classification and regression trees.

This is the reason why BN is used in software development projects for effort estimation (Bibi and Stamelos, 2004; Mendes et al., 2012), reliability evaluation (Si et al., 2014), quality prediction (Jeet et al., 2011; Schulz et al., 2010), risk assessment (Chatzipoulidis et al., 2015; Lee et al., 2009), testing-effort estimation (Wooff et al., 2002), and so on. But only a few uses of BN have been reported in agile software development projects. A Systematic Literature Review (Usman et al., 2014) confirms these results. The most commonly used assessment methods in agile software development are expert judgment, planning poker and use case points, even though these methods do not result in good prediction accuracy.

Hearty et al. (2009) describe the use of a Dynamic Bayesian Network causal model for the Extreme Programming (XP) methods. A Dynamic Bayesian Network (DBN) is a BN expanded by a temporal dimension, so that the changes over time can be modelled. The changes of XP's key Project Velocity metric are modelled to make effort estimation and risk assessments. The model is validated against the real-world XP project.

Abouelela and Benedicenti (2010) also use BN for modelling XP software development process. This model consists of two models: one for the estimation of the project duration, and the other for the estimation of the expected defect rate. The estimations are based on the use of three XP practices: Pair Programming, Test Driven Development and Onsite Customer. The model is validated against two XP projects.

Perkusich et al. (2013) use a BN model for Scrum project modelling to provide information to Scrum Master for problem detection. It is validated with data from ten different scenarios.

Nagy et al. (2010) create another BN model to assist a project manager in decision making. This BN model evaluates several key factors which influence the development of software in order to detect problems as early as possible. The model is not validated.

Due to a small number of papers on the use of the BN model in agile software development projects, we also examine the use of BN in iterative development. Torkar et al. (2010) depict the use of a DBN for test effort estimation. The DBN model is based on process and resource measurements. Process measurement is related to software activities such as development and support. Resource

measurement is related to assets such as people, tools and equipment. The DBN model consists of two sub-models: the test process overall effectiveness model and the test effort model. The authors use data collected from two industrial projects to validate the DBN model.

The Systematic Literature Review of Usman et al. (2014) shows that only Extreme Programming (XP) and Scrum Methods have been investigated in the estimation studies. However, development teams usually do not strictly stick to all the practices of the selected methodology. They mostly choose the subset of agile practices which is suitable for the specific project (Williams, 2010). Consequently, the estimation model should not depend on the selected agile practices and methods.

We are not aware of any research that attempts to use a BN model to predict agile project effort, regardless of the selected agile method and without impact on agility. In all the above mentioned papers, validation processes (if performed) have one common characteristic: all the BN models are validated against a small number of projects (only one or two). This is the reason we take a project task as the smallest estimation and validation record – there are 160 project tasks in our proposed model. Moreover, our BN model is also validated against a small number of project tasks.

Some of the mentioned BN models are relatively big (Perkusich et al., 2013). We want to create a model which will have a satisfactory accuracy with a minimal set of input data. DBN models are usually smaller, but they are unable to predict effort in the first iteration (Hearty et al., 2009; Torkar et al., 2010). It is extremely important that the BN model can be used as early as possible.

To summarize, the existing BN models do not estimate task effort. Besides, none of the above described models meets all the following requirements:

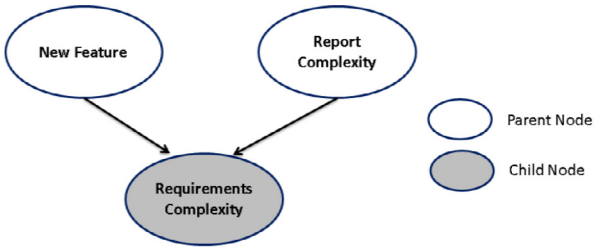
- Suitability for agile development, regardless of used agile methods and/or practices.
- Minimal set of input parameters, provided that the method predicts with at least 75% accuracy.
- Possibility of using the BN model at the start of the project.
- Validation based on a larger sample size (not just a few samples).

3. Bayesian network

A Bayesian network (BN) is a graphical model that describes probabilistic relationships between causally related variables.

The BN is formally determined by the pair $BN = (G, P)$, where G is a directed acyclic graph (DAG), and P is a set of local probability distributions for all the variables in the network. A directed acyclic graph $G = (V(G), E(G))$ consists of a finite, nonempty set of tuples $V(G) = \{(s_1, V_1), (s_2, V_2), \ldots, (s_n, V_n)\}$ and a finite set of directed edges $E(G) \subseteq V(G) \times V(G)$ (see Fig. 1). Nodes $V_1, V_2, \ldots, V_n$ correspond to random variables $X = (X_1, \ldots, X_n)$, that can take on a certain set of values s_i (depending on the problem being modelled). The terms variable and node will be used interchangeably in this paper. The edges $E(G) = \{e_{i,j}\}$ represent dependencies among variables. A directed edge $e_{i,j}$ from V_i to V_j for $V_i, V_j \in V(G)$ shows that V_i (parent node) is a direct cause of V_j (child node).

Each variable X_i has a joint probability distribution $P(X_i \mid \text{parent}(X_i))$ which shows the impact of a parent on a child. If X_i has no parents, its probability distribution is unconditional, otherwise, it is conditional. The probability distribution of variables in a BN must satisfy the Markov condition, which states that each variable X_i is independent of its nondescendents, given its parents in G (Charniak, 1991). The BN decomposes the joint probability distribution $P(X_1, \ldots, X_n)$ into a product of conditional probability distributions for each variable given its parents in:



NPT for Node New Feature	
No	Yes
0,7	0,3

NPT for Node Report Complexity		
Low	Medium	High
0,2	0,3	0,5

NPT for Node Requirements Complexity				
New Feature	Report Complexity	Low	Medium	High
No	Low	0,7	0,2	0,1
No	Medium	0,3	0,5	0,2
No	High	0,1	0,7	0,2
Yes	Low	0,3	0,5	0,2
Yes	Medium	0,2	0,5	0,3
Yes	High	0,1	0,2	0,7

Fig. 1. The BN model and the associated Node Probability Tables (NPTs).

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid \pi(x_i)),$$

where $\pi(x_i)$ stands for the set of parents of x_i, or, in other words, the set of nodes that are directly connected to x_i via a single edge.

An example of a simple BN is shown in Fig. 1, together with the associated Node Probability Tables (NPTs). Each row in the NPT represents a conditional probability distribution and, therefore, its values sum up to 1. The nodes "New Feature" and "Report Complexity" are root nodes (nodes without parents), so a priori probabilities are defined for them. Conditional Probabilities for all possible combinations of outcomes of its parents are defined for the node "Requirements Complexity". For example, for the relative probability of "Requirements Complexity" being 'High', conditional on "New Feature" being 'Yes', and "Report Complexity" being "Medium", the Conditional Probability is 0,3.

The BN is mainly used for presenting ambiguities in various domains[1] in a simple and easily understandable way, for making predictions as well as diagnostics, for computing the probabilities of occurrence of an event, and for updating the calculations according to evidences (Fenton and Neil 1999; Fenton et al. 2008).

4. The proposed BN model

4.1. The BN model requirements

Agile methods avoid the formalisms of traditional specification and design techniques. The downside of this is a lack of specification metrics for project planning. At the same time, agile project managers have to plan their projects as any other traditional project manager. The main purpose of this paper is to build a BN which can help agile project managers predict project effort. The proposed BN model should meet the following conditions:

[1] A query "Bayesian" in the Scopus database (all areas, all time periods) results in 123.139 papers (Scopus, 2016a). Another query "Bayesian" in the Scopus database that excludes computer science, engineering and mathematics papers (all time periods), results in 48.602 papers (Scopus, 2016b).

- Applicability for any agile method.
- Minimization of impact on agility:
  - Input data must be simple to collect.
  - Model must be as small and simple as possible.
- Predictability of effort so that (a lack of) success can be evaluated.
- Learnability from newly entered data.
- Possibility of use as early as possible, during planning phase, before actual start.
- Ability to process different data types (Boolean, rank, integer, etc.).

When building a BN model, it is essential to define the problem which is to be solved. Initially, the intention was to build a BN model for sprint (iteration) effort prediction. The model was planned to use the existing database of software projects for model validation. These are the projects of a micro software company which has been using agile methods for several years now. The project manager (scrum master) has two databases:

- The list of software entities (with their complexity classifications) extracted for each task from the project log (Celar et al., 2012).
- The list of knowledge and skills calculated for each developer, including motivation and experience, classified into 5 levels (from 1, very low level, to 5, very high level), and updated twice a year (Celar et al., 2014).

When planning new tasks, the project developers and the project manager try to find the best solution for both sides: developers' ideas and project productivity. When working on a task, the developers record their working hours and the types of activities (that takes only 1–2 minutes at the end of a workday). So, they always know the real task and project status. Thanks to this short developers' activity, the project manager receives very valuable information. Consequently, the authors realized very early in the process that it was convenient to build a BN model for task effort prediction because:

- It is easier to collect input data.
- Input data are less complex.
- It is easier for a project manager to estimate a node value (e.g., whether the complexity of a report is "low", "medium" or "high").
- It is easy to predict the iteration duration based on each developer's prediction effort.

4.2. Measures to assess the accuracy of BN models

Some uncertainty is always included in effort prediction because each project is unique – there are no two projects with same requirements, priorities, technology or developers. Uncertainty decreases significantly as new knowledge is obtained during the project. Anyhow, at the beginning, when a lot of information is unknown, effort estimation is especially difficult. As the project progresses, estimations become increasingly accurate (see Fig. 2), but in the early phase of the project the cone of uncertainty shows that the actual value varies from 40 to 250% (McConnell, 2006) or from 60 to 160% (Kan, 2002) compared to the estimation. Models using BN for effort prediction declare an accuracy range from 14 to 100% (Radlinski, 2010). A prediction accuracy of 80% (significantly over all predictions) or more is usually satisfactory.

The Mean Magnitude of Relative Error (MMRE) and the Prediction at Level m (Pred. (m)) are the two most commonly used metrics to assess the accuracy of prediction in software estimation. These measures are also used in this research.

But, in our research, there is only one prediction per one task. Therefore, other statistical measures are also used to assess the accuracy of this model: Accuracy, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Relative Absolute Error (RAE) and Root Relative Squared Error (RRSE). These measures are chosen due to their simplicity of use and because of their application areas (Hernandez-Orallo et al., 2012; Kim et al., 2014; Sarwar et al., 2001).

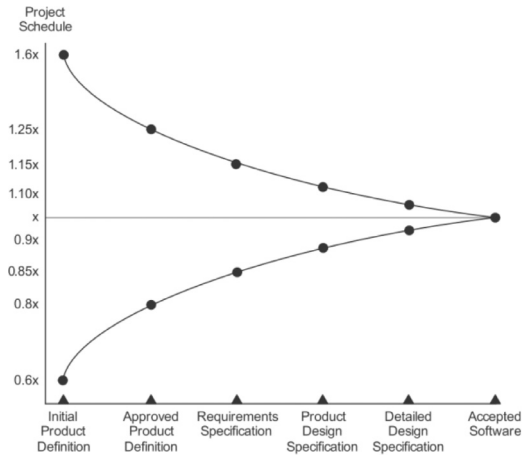


Fig. 2. The cone of uncertainty (Kan, 2002).

MMRE is the average of the Magnitude of Relative Errors (MREs) calculated over all the reference tasks:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i.$$

MRE represents the normalized measure of deviations between the actual and the estimated values:

$$MRE = \frac{|y_i - f(x_i)|}{y_i}.$$

Although MMRE is the most frequently used prediction accuracy measure, there are criticisms of its applicability. Kitchenham et al. (2001) suggest that MMRE is more suitable for goodness of fit statistics than for evaluation of prediction. MMRE is the measure of standard deviation (spread) of a variable x_i (x_i = predict value/actual value), while Pred. (m) is the measure of peakedness (kurtosis) of a variable x. MMRE prefers models that predict estimates below the mean, and it is also sensitive to outliers (Foss et al., 2003; Korte and Port, 2008). A sensitivity to outliers allows MMRE to detect whether the model occasionally tends to be very inaccurate.

Pred. (m) measures the percentage of estimates that are within m percent of the actual values. It is usually set to $m = 25$. Pred. (25)% detects what percentage of estimates is within a tolerance of 25%.

Accuracy is the percentage of the correctly classified instances over the total number of instances. The accuracy can range from 0 to 100%.

MAE is the average of absolute values of prediction errors, given by:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |f(x_i) - y_i|,$$

where n is the number of the predictions, $f(x_i)$ is a predicted value, and y_i is an observed value. All the errors are weighted equally due to the linear score.

RMSE is another measure of deviation between the predicted $f(x_i)$ and the real value y_i:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (f(x_i) - y_i)^2}.$$

Large errors are weighted more heavily because the errors are averaged after they are squared. The error variance can be detected if RMSE and MAE are used together. The variation is greater if the difference between them is larger. If all the errors have the same magnitude, MAE = RMSE, otherwise, RMSE > MAE.

Both MAE and RMSE are useful for the comparison of the prediction errors of different models for a particular variable and not for the comparison between variables. They show errors in the same unit and scale as the parameter itself, so they are scale-dependent.

We also include measures that can be used for the comparison of models whose errors are measured in different units. Such measures include:

- Relative Absolute Error (RAE):

$$RAE = \sum_{i=1}^{n} (|f(x_i) - y_i|) \Big/ \sum_{i=1}^{n} (|\bar{y}_i - y_i|).$$

- Root Relative Squared Error (RRSE):

$$RRSE = \sqrt{\sum_{i=1}^{n} (f(x_i) - y_i)^2 \Big/ \sum_{i=1}^{n} (\bar{y}_i - y_i)^2},$$

$$RRSE = \sqrt{\sum_{i=1}^{n} (f(x_i) - y_i)^2 \Big/ \sum_{i=1}^{n} (\bar{y}_i - y_i)^2},$$

where $\bar{y}_i$ is the mean value of y_i.

5. The BN building process

The BN building process can be divided into two steps: a definition of the elements of the set G (the DAG structure definition) and a definition of the elements of the set P (parameter estimation). Both, the DAG structure definition and the parameter estimation, can be either purely expert based, or learned purely from the data. Some combination of expert knowledge and empirical data can also be used. The topology of BN for effort prediction is mostly expert defined (Radlinski, 2010), but the final structure of BN is mainly built based on the combination of expert knowledge and inference algorithms (Misirli and Bener, 2014). The BN model presented in this paper is based on quantitative historical data from a software project database and on the authors' expert knowledge. All the authors are experienced project managers, and two of them also have experience in the construction of BN.

In this research, the definition of set G is separated in three sub-tasks:

- Identification of nodes V_1, V_2,..., V_n in the network (variables in the problem).
- Identification of all the elements of set s_i, i.e., the definition of all the possible outcomes for each node (values that variables can take).
- Identification of set $E = \{e_{i,j} \mid V_i, V_j \in V(G)\}$, i.e., a definition of all the network edges which show the dependencies of the variables.

The set P (V_1, V_2 ..., V_n) is defined when:

- A priori probabilities for nodes without parents (root nodes) are defined.
- Conditional probabilities for all nodes with parents, and for all possible combinations of outcomes of their parents, are defined. A priori probabilities for nodes with parents are defined through an associated table of joint probability distributions, and through a priori expectations of their parents. Therefore, it is superfluous to define explicitly a priori probabilities for nodes with parents.

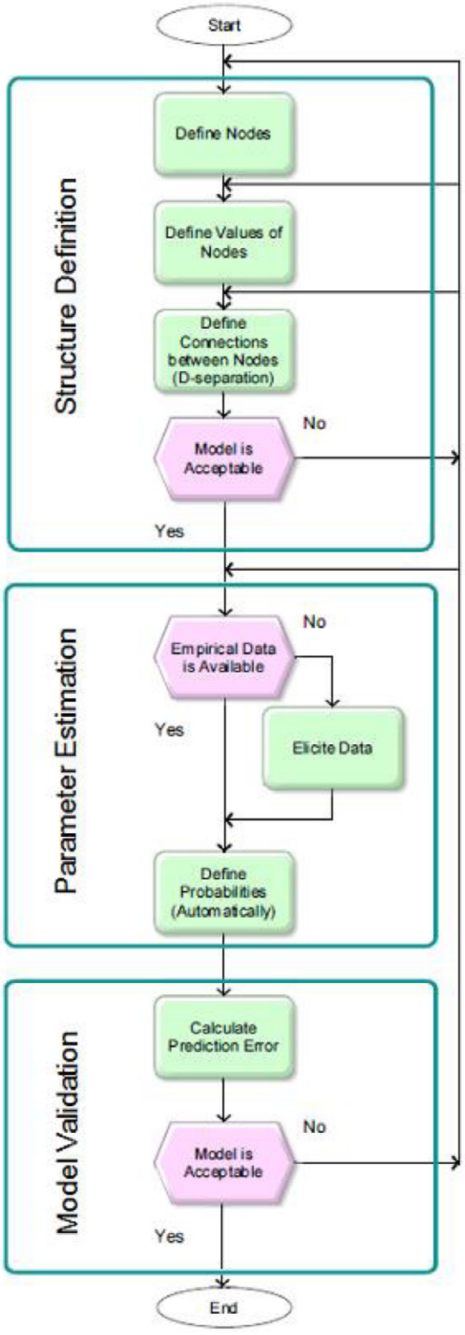


Fig. 3. The BN building process.

For the proposed BN model, these probabilities are obtained automatically from the data. The BN building process is depicted in Fig. 3. It took three iterations through the process to build the final version of the BN models.

5.1. Criteria collection (Node definition)

The authors strive to establish a minimum set of criteria (cost drivers) needed for evaluation. There is a great diversity in the use of cost drivers and no consensus exists regarding which criteria should be used in a specific context (Usman et al., 2014). The most widely used cost drivers in agile development are: task size, skills and experience of developers. These cost drivers are also used in the proposed model, but they are not sufficient for an accurate assessment.

This step focuses on gathering the criteria specific for software effort estimation from the existing literature (Abouelela and Benedicenti, 2010; Hearty et al., 2009; Mendes et al., 2012; Misirli and Bener, 2014; Nagy et al., 2010; Perkusich et al., 2013; Radlinski, 2010; Torkar et al., 2010; Usman et al., 2014).

The survey of BN models for software effort prediction (Radlinski, 2010) shows that the criteria can be grouped in four categories based on the measured characteristics:

- Project scope – determined by the use of various measures, e.g., use cases or user stories, function points, lines of codes, requirements, etc.
- Other project characteristics – considered in relation to type, complexity and stability of the project.
- Staff factors – which include, among others, developer skills, experience and motivation.
- Process characteristics – considered in relation to the organization and the maturity of the development process.

The elements of set *V* (BN nodes) are selected by applying a Goal Question Metric (GQM) approach to the collected criteria (Basili et al., 1994; Differding et al., 1996). The GQM plan consists of a goal and a set of questions and measures. The plan describes precisely why the measures are defined and how they are going to be used. The asked questions help to identify information required to fulfil the goal. The measures define the data to be collected to answer the questions.

The most important goal of the task effort prediction is to determine the time needed for task completion. Hence, the first element of set *V* is defined: *Working Hours*. Task effort depends on the complexity of requirements and on the developer skills (including motivation and experience). So, the next two elements of *V* are defined: *Requirements Complexity* and *Developer Skills*. The effort of the task depends largely on whether the programmer is familiar with this type of task or he has to use new technologies and new knowledge. Thus, the next element is: a *New Task Type*. The complexity of the requirements depends on the number and the complexity of reports, user interfaces (forms) and functions that should be created in a task, as well as on the quality of the requirements specifications. In the first iteration, set *V* is completed by elements: *Form Complexity, Report Complexity, Function Complexity* and *Specification Quality*.

The GQM approach ensures that all the relevant domain variables are included. The authors also checked and assured themselves that the variables were named conveniently.

5.2. Node values

To fully define a set of tuples $V(G) = \{(s_1, V_1), \ldots, (s_n, V_n)\}$, it is necessary to define s_i, the set of all possible values for each V_i.

To define node values, the authors first checked the past data. As mentioned before, the authors have access to database of 160 tasks, but agile development software projects are famous for very limited documentation. The existing data are usually unstructured, meaning that they are not suitable for direct use in the BN. Even the structured data should be ranked. For example, a variable *Working Hours* expresses the number of hours spent on an individual task. The range of values is too large in terms of too many potential outcomes. To simplify possibilities, the outcome values should be intervals instead of point values. Therefore, a new node *Working Hours Classification* is added, with possible outcomes ranked in five intervals (Table 1). Instead of adding a new node, the values of *Working Hours* can be split in five intervals. Furthermore, the authors hope that it will be possible to refine values in more intervals after the model is used, and more data became available. In that case, it will be easier to change the *Working Hours Classification* only.

Table 1
Nodes description.

Node name	Description
Form Low_No	Number of simple user interfaces (forms)
Form Medium_No	Number of moderate complexity user interfaces (forms)
Form High_No	Number of complex user interfaces (forms)
Function Low_No	Number of simple functions
Function Medium _No	Number of moderate complexity functions
Function High _No	Number of complex functions
Report Low_No	Number of simple reports
Report Medium _No	Number of moderate complexity reports
Report High _No	Number of complex reports
Form Complexity	Total rating of user interface (form) complexity
Function Complexity	Total rating of function complexity
Report Complexity	Total rating of report complexity
Specification Quality	Quality of specification
New Task Type	Type of task (new or familiar one)
Requirements Complexity	Overall rating of requirements complexity
Developer Skills	Overall rating of developer experience, motivation and skills
Working Hours	Number of hours spent on the task
Working Hours Classification	Intervals of spent working hours: 0–2 h – very simple task (56 instances) 2,1–10 h – simple task (70 instances) 10,1–25 h – moderate task (22 instances) 25,1–40 h – complex task (6 instances) >40 h – very complex task (6 instances)

The values of nodes are defined in two steps:

- The first step defines the types of the selected variables and identifies the values for each variable. Although BN allows the use of both discrete and continuous variables, in this paper we use discrete values, because the experimental data are discrete, and because the available BN tools require the discretization of the continuous variables.
- The second step is extremely time-consuming. An experienced project manager in agile development checks all the projects, evaluates the unstructured data and creates a database suitable for BN. As task evaluation is time-consuming, tasks are processed in batches: first 40, then 50, and finally 70 tasks. All the values in the newly created database are checked for rank and accuracy. In some cases, it is necessary to go back to the first step and refine the values of the nodes.

5.3. DAG structure construction (Node connections)

The GQM approach used for the definition of the elements of set *V* is also used for the definition of set *E*. The causal relationships between the nodes are built based on variables and measures selected by using GQM. The building process includes d-separation (d-separation dependencies are used to identify variables influenced by evidence coming from other variables in the BN), as well as a new node definition.

For example, a variable *Report Complexity* represents the complexity of the reports that should be created in a task. The variable can take one of three states (low, medium or high), and its value can be estimated by the project manager. Instead of that, three new parent nodes (variables) are added: *Report Low_No* (number of simple reports), *Report Medium_No* (number of medium complexity reports) and *Report High_No* (number of complex reports). This is important because with parent nodes the state of variable *Report Complexity* is defined automatically, thus avoiding a situation where one project manager estimates the same 10 simple reports once as low, and at other times as moderately complex. Consistency significantly affects the accuracy of prediction of the BN model.

For the same reason, parent nodes are also added to nodes *Form Complexity* (defines complexity of user interfaces) and

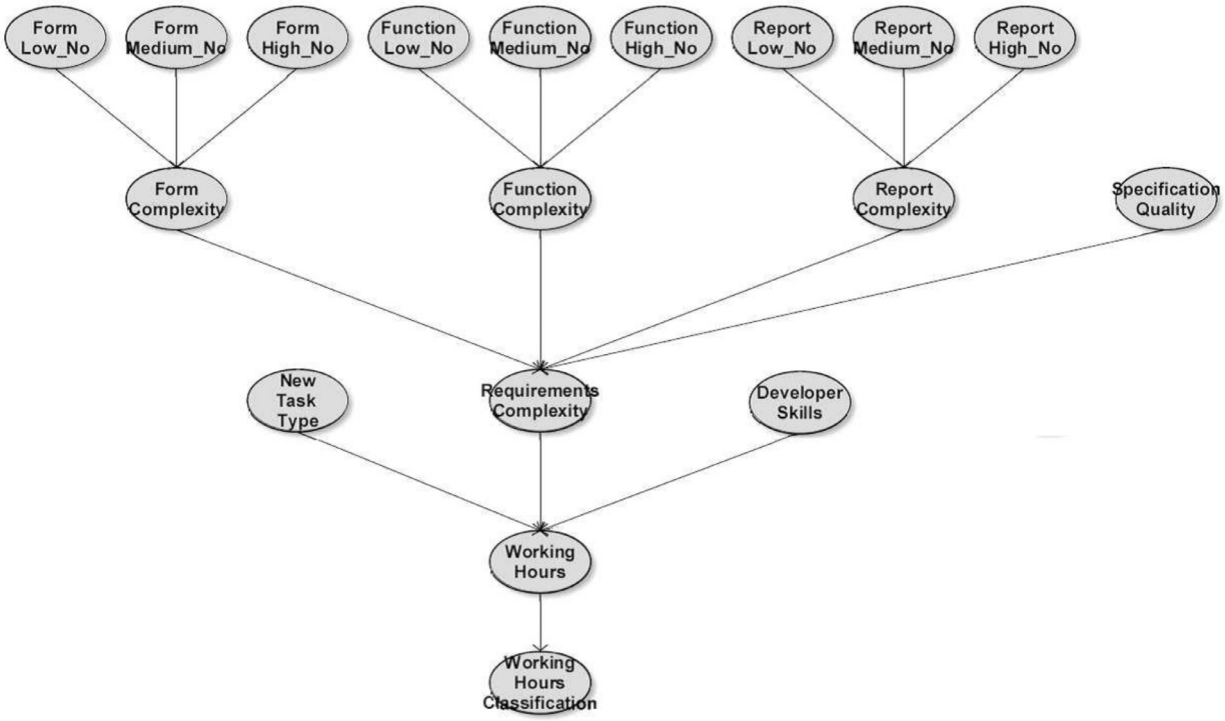


Fig. 4. The BN model.

Table 2
Empirical data prepared for use in the BN model (Part).

Task ID	New Task Type	Specification Quality	Form Low_No	Form Medium_No	Form High_No	Function Low_No	Function Medium_No	Function High_No	Report Low_No	Report Medium_No	Report High_No	Developer Skills	Working Hours
1	Yes	2	2	0	2	0	10	0	0	0	0	2	16,5
2	Yes	1	3	2	0	0	2	0	0	0	2	4	9
3	Yes	4	0	2	2	3	3	3	0	0	0	3	8,5
4	No	4	5	0	0	5	0	0	0	0	0	2	28
5	Yes	3	0	0	5	0	0	0	0	0	5	4	46,5
6	No	3	0	1	0	0	0	0	1	0	0	2	1
7	No	2	0	1	0	0	1	0	1	0	0	3	1,5
8	Yes	3	2	2	5	0	3	3	1	0	0	3	9
9	Yes	3	0	0	0	0	0	0	0	4	0	2	3,75
10	No	4	1	0	0	0	2	0	0	0	1	2	10
11	No	2	0	1	1	0	1	0	0	0	0	3	0,5
12	Yes	5	1	1	1	0	3	0	0	0	0	2	15,5

Function Complexity (defines complexity of function). The values of both nodes can be expressed as *high, medium* or *low*. Consequently, the *Form Complexity* is a child node of *Form Low_No* (number of simple forms), *Form Medium_No* (number of medium complexity forms) and *Form High_No* (number of complex forms). On the other side, *Function Low_No* (number of simple functions), *Function Medium_No* (number of medium complexity functions) and *Function High_No* (number of complex functions) are parents of the *Function Complexity* node.

The complexity of the reports as well as the complexity of the forms and functions is defined on the basis of the elements to be constructed, their number, and their comparison with historical data on similar elements (analogy). The report evaluation is also influenced by the database query complexity used to obtain the result. The assessment of the function complexity also depends on the complexity of the processing algorithm.

The specification quality is determined by a level of requirements decomposition: definition of technical demands and business clarity.

The estimation of skills and knowledge as well as experience and motivation of each developer is rated by the Personal Capability Assessment Method (Celar et al., 2014) and then ranked in to one of 5 grades. The evaluation of a developer is performed once or twice a year.

A new iteration of the model building process starts each time when a new node is added. A list of all the nodes with explanations of their meaning is given in Table 1. The final topology is shown in Fig. 4.

5.4. Parameter estimation

Conditional and a priori probabilities are learned from the data using the WEKA[2] machine learning suite.

As we already mentioned, the data used in this research originate from agile projects of a small software company. These data are not suitable for direct use in the BN model. They must be prepared, but, as the process of preparation is time-consuming, the data are separated in three datasets. The first dataset consists of 40 tasks, the second of 50 tasks, and the third of 70 tasks. Tasks are

[2] Waikato Environment for Knowledge Analysis (WEKA) 3.6.11, http://www.cs.waikato.ac.nz/ml/weka/.

grouped chronologically on the basis of creation time. Grouping is made neither by size, nor by complexity, nor according to the developer who performs the task. All the datasets include tasks of different duration and complexity, created by different developers. A set of empirical data for the twelve tasks is shown in Table 2.

Empirical data are not available for all the nodes. Nodes *Form Complexity, Function Complexity, Report Complexity, Requirements Complexity* and *Working Hours Classification* are added to simplify the possible outcomes, as well as to provide better model accuracy. The values of *Working Hours Classification* are ranked in five non-linear intervals based on the authors' experience. The manual definition of NPTs can be a lengthy and error prone process. Consequently, the values of these nodes are evaluated on the basis of empirical values of their parents. The probabilities are automatically learned both for empirical and added nodes.

An example of a table with complete data for parameter estimation in BN model is shown in Table 3. It consists of empirical data from Table 2, completed with the data estimated by the authors.

5.5. BN model validation

The model validation is performed using empirical data. WEKA provides a k-fold cross-validation and a summary statistics (prediction accuracy, MAE, RMSE) which are used to verify the accuracy of the generated model. The Weka error statistics are normalized. The predicted distribution for each class is matched against the expected distribution for that class. All the mentioned Weka errors are computed by summing over all classes of an instance, not just a true class (WEKA, 2007a; WEKA, 2007b).

In this case, a 10-fold cross-validation is used. The dataset is randomly divided into 10 equally sized subsets. Of these 10 subsets, one is taken as the validation dataset, and the other nine sets are used as training data. Each of the nine training datasets is compared with a validation dataset to calculate the percentage of the model accuracy. The cross-validation process is repeated ten times, each of the ten subsets being used exactly once as a validation set. The results from all the 10 trials are averaged.

The prediction accuracy for all the datasets is shown in Table 4. The accuracy is above 90% for all datasets. For a dataset of 160 instances, only the effort of one task is wrongly classified. The MAE values indicate that the expected effort will be within 2.6% of the true effort for the last set of data. Small differences between the MAE and the RMSE values indicate that the error variance is relatively small.

The Pred. (m) and the MMRE metrics are also applied to this dataset. Since the BN model estimates effort as a set of probability distributions for all possible classes, a Conversion method (Pendharkar et al., 2005; Mendes, 2008; Tierno, 2013) is used to obtain the estimated effort as a discrete value. The class probabilities should be normalized, so that their sum equals one. The estimated effort is then given by:

$$Effort = \sum_{i=1}^{n} \rho_{classi} \mu_{classi},$$

where μ_{classi} is the mean of class i, and ρ_{classi} is its respective class probability.

The MMRE values suggest that the prediction error is relatively constant, with no occasional large deviations.

The predictions for all the sets are within 25% of the actual values. Even in the case of the stricter criterion ($m = 10$), 90% of the estimates for all the sets remain within a 10% tolerance. Moreover, we should note that this is the result of the set with minimum number of instances, and the BN prediction accuracy grows with the growth of number of instances.

Table 3
Complete data used for parameter learning (12 task example).

Task ID	New Task Type	Specification Quality	Form Low_No	Form Medium_No	Form High_No	Form Complexity	Function Low_No	Function Medium_No	Function High_No	Function Complexity	Report Low_No	Report Medium_No	Report High_No	Report Complexity	Requirements Complexity	Developer Skills	Working Hours	Working Hours Classifcation
1	Yes	2	2	0	2	Medium	0	10	0	Medium	0	0	0	Low	Medium	2	16,5	10,1-25
2	Yes	1	3	2	0	Medium	0	2	0	Medium	0	0	2	High	Medium	4	9	2,1-10
3	Yes	4	0	2	2	High	3	3	3	High	0	0	0	Low	High	3	8,5	2,1-10
4	No	4	5	0	0	Low	5	0	0	Low	0	0	0	Low	Low	2	28	25,1-40
5	Yes	3	0	0	5	High	0	0	0	Low	0	0	5	High	High	4	46,5	>40,1
6	No	3	0	1	0	Low	0	0	0	Low	1	0	0	Low	Low	2	1	0-2
7	No	2	0	1	0	Low	0	1	0	Medium	1	0	0	Low	Low	3	1,5	0-2
8	Yes	3	2	2	5	High	0	3	3	High	1	0	0	Low	High	3	9	2,1-10
9	Yes	3	0	0	0	Low	0	0	0	Low	0	4	0	Medium	Low	2	3,75	2,1-10
10	No	4	1	0	0	Low	0	2	0	Medium	0	0	1	High	Medium	2	10	2,1-10
11	No	2	0	1	1	Medium	0	1	0	Medium	0	0	0	Low	Medium	3	0,5	0-2
12	Yes	5	1	1	1	Medium	0	3	0	Medium	0	0	0	Low	Medium	2	15,5	10,1-25

Table 4
Results.

Number of tasks	40 (first)	50 (next)	70 (last)	90 (40+50)	160
Accuracy (Correctly classified instances)	90%	96%	97.14%	96.67%	99.375%
MAE	0.1065	0.0533	0.0531	0.0469	0.026
RMSE	0.199	0.1301	0.1127	0.117	0.065
RAE	35,28%	24,23%	19,89%	17,69%	9,71%
RRSE	51,27%	40,09%	31,03%	32,31	17,81%
Pred. (25)%	100%	100%	100%	100%	100%
Pred. (10)%	90%	100%	100%	100%	100%
MMRE	12.80	3.29	4.30	7,69	6.21

These good results (Table 4) make this simple model applicable in practice. According to the cone of uncertainty, the results are better than expected. The high accuracy of the model is confirmed by comparison with the results listed in the literature (Radlinski, 2010).

Such a good prediction accuracy is mainly based on the following:

- The BN model outcomes are probability distributions for only five intervals. This decreases the prediction precision because all the values in an interval are treated equally. For example, values 45 and 61 from the interval '>40 hours' have the same probabilities.
- A priori and conditional probabilities are automatically attained from the experimental data, which is more reliable than elicitation from scratch. The estimation accuracy of these probabilities has a notable effect on the outcome quality, and either a pessimistic or an optimistic approach can spoil the results.
- The consistency in assessment significantly increases the accuracy of the prediction. Most experimental data were not suitable for direct application to the BN model, e.g., values that should be ranked. Two of the authors have independently evaluated all the data to make sure that the values are consistently assessed.

6. Conclusions and future work

This paper develops a BN model for effort prediction in agile software development projects.

The proposed model is relatively small and simple and all the input data are easily elicited, so that the impact on agility is minimal. The model predicts task effort, and it is independent of agile methods used. It is also suitable for use in the early project phase.

The model is validated using a database of 160 tasks from real agile projects. The prediction accuracy is measured by the percentage of correct over all predictions. The model results in very good accuracy: only one misclassified value. Pred. ($m = 25$) equals 100% – all predictions are classified within 25% tolerance. The MMRE values show that there are no occasional large estimation errors. All the other statistical metrics used in this research support these results.

This BN model is presently used in one software company, and the project manager considers it very useful.

The proposed BN model is currently being expanded with a new subnet regarding the existing node *Developer Skills* and with a new outcome variable/node *Product Quality*. We plan to use this model for quality prediction of software products in the early software project phase.

Acknowledgments

This work has been supported in part by the PIVIS project (1904-10), technological project at FESB funded by the enterprise MIB PIVAC, Vrgorac, Croatia, the Croatian Science Foundation under the project INSENT Innovative Smart Enterprise (1353), Zagreb, Croatia, and the Program of Technological Development, Research and Application of Innovations of Split-Dalmatia County (1012-14), Split, Croatia. Special thanks belong to Prof. Sanda Halas for her language advice.

References

Abouelela, M., Benedicenti, L., 2010. Bayesian network based XP process modelling. Int. J. Softw. Eng. Appl. 1 (3), 1–15.
Basili, V.R., Caldiera, G., Rombach, H.D., 1994. The goal question metric approach. In: The Encyclopedia of Software Engineering, 1. John Wiley & Sons, New York, USA, pp. 469–476.
Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., et al., 2001. The Agile Manifesto, http://www.agileAlliance.org/.
Bibi, S., Stamelos, I., 2004. Software process modeling with bayesian belief networks. In: Proceedings of the 10th International Software Metrics Symposium (Metrics 2004). Chicago, USA.
Borade, J.G., Khalkar, V.R., 2013. Software project effort and cost estimation techniques. Proceedings of the International Journal of Advanced Research in Computer Science and Software Engineering 3 (8), 730–739. ISSN: 2277 128X.
Celar, S., Turic, M., Vickovic, L., 2014. Method for personal capability assessment. In: Proceedings of the 22nd Telecommunications Forum Agile Teams Using Personal Points. Beograd, Serbia. IEEE, pp. 1134–1137.
Celar, S., Vickovic, L., Mudnic, E., 2012. Evolutionary measurement-estimation method for micro, small and medium-sized enterprises based on estimation objects. Adv. Prod. Eng. Manag. 7 (2), 81–92.
Charniak, E., 1991. Bayesian networks without tears: making Bayesian networks more accessible to the probabilistically unsophisticated. AI Mag. 12 (4), 50–63.
Chatzipoulidis, A., Michalopoulos, D., Mavridis, I., 2015. Information infrastructure risk prediction through platform vulnerability analysis. J. Syst. Softw. 106, 28–41.
Cohn, M., 2005. Agile Estimating and Planning, 3–4. Prentice Hall, Upper Saddle River, USA, pp. 43–47.
Differding, C., Joisl, B., Lott, C. M., 1996. Technology Package for the Goal Question Metric Paradigm, Technical Report 281/96, University of Kaiserslautern, Germany.
Dragicevic, S., Celar, S., 2013. Method for elicitation, documentation and validation of software user requirements (MEDoV). In: Proceedings of the 18th IEEE International Symposium on Computers and Communications (ISCC). Split, Croatia,.
Dragicevic, S., Celar, S., Novak, L., 2011. Roadmap for requirements engineering process improvement using BPM and UML. Adv. Prod. Eng. Manag. 6 (3), 221–231.
Dragicevic, S., Celar, S., Novak, L., 2014. Use of method for elicitation, documentation and validation of software user requirements (MEDoV) in agile methods. In: Proceedings of 6th International Conference on Computational Intelligence, Communication Systems and Networks/(CICSyN). Tetovo, Macedonia. IEEE.
Fenton, N., Hearty, P., Neil, M., Radliński, Ł., 2008. 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, USA, pp. 1–25.
Fenton, N., Neil, M., 1999. A critique of software defect prediction models. IEEE Trans. Softw. Eng. 25 (5), 675–689.
Foss, T., Stensrud, E., Kitchenham, B., Myrtveit, I., 2003. A simulation study of the model evaluation criterion MMRE. IEEE Trans. Softw. Eng. 29 (11), 985–995. doi:10.1109/TSE.2003.1245300.
Hearty, P., Fenton, N., Marquez, D., Neil, M., 2009. Predicting project velocity in XP using a learning dynamic Bayesian network model. IEEE Trans. Softw. Eng. 35 (1), 124–137.
Helmy, W., Kamel, A., Hegazy, O., 2012. Requirements engineering methodology in agile environment. IJCSI 9 (5), 293–300 No. 3, ISSN (Online): 1694–0814.
Hernandez-Orallo, J., Flach, P., Ferri, C., 2012. A unified view of performance metrics: translating threshold choice into expected classification loss. J. Mach. Learn. Res. 13 (1), 2813–2869.
Jeet, K., Bhatia, N., Minhas, R.S., 2011. A Bayesian network based approach for software defects prediction. ACM SIGSOFT Softw. Eng. Notes 36 (4), 1–5.
Jorgensen, M., 2014. What we do and don't know about software development effort estimation. IEEE Softw. 31 (2), 37–40.
Jørgensen, M., 2013. Relative estimation of software development effort: it matters with what and how you compare. IEEE Softw. 30 (2), 74–79. doi:10.1109/MS.2012.70.
Jorgensen, M., 2010. Selection of strategies in judgment-based effort estimation. J. Syst. Softw. 83, 1039–1050. doi:10.1016/j.jss.2009.12.028.
Kan, S.H., 2002. Metrics and Models in Software Quality Engineering, 2nd ed. Addison-Wesley Longman Publishing Co., Inc., Boston, USA ISBN: 0201729156.
Kim, S.T., Hong, S.R., Kim, C.O., 2014. Product attribute design using an agent-based simulation of an artificial market. Int. J. Simul. Model. 13 (3), 288–299.
Kitchenham, B.A., Pickard, L.M., MacDonell, S.G., Shepperd, M.J., 2001. What accuracy statistics really measure. IEEE Proc. Softw. 148 (3), 81–85. doi:10.1049/ip-sen:20010506.
Korte, M., Port, D., 2008. Confidence in software cost estimation results based on MMRE and PRED. In: Proceedings of the 4th international workshop on Predictor models in software engineering, pp. 63–70. doi:10.1145/1370788.1370804.

Lee, E., Park, Y., Shin, J.G., 2009. Large engineering project risk management using a Bayesian belief network. Expert Syst. Appl. Int. J. 36 (3), 5880–5887.
McConnell, S., 2006. Software Estimation: Demystifying the Black Art. Microsoft Press, WA, USA.
Mendes, E., 2008. The use of Bayesian networks for web effort estimation: further investigation. In: Proceedings of the Eighth International Conference on Web Engineering, Proceedings of ICWE'08, pp. 203–216.
Mendes, E., Abu Talib, M., Counsell, S., 2012. Applying knowledge elicitation to improve web effort estimation: a case study. In: Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference, COMPSAC '12. Izmir, Turkey, pp. 461–469.
Misirli, A.T., Bener, A.B., 2014. A Mapping Study on Bayesian Networks for Software Quality Prediction. RAISE'14 PROGRAM, Hyderabad, India.
Nagy, A., Njima, M., Mkrtchyan, L., 2010. A Bayesian based method for agile method software development release planning and project health monitoring. In: Proceedings of the 2010 IEEE International Conference on Intelligent Networking and Collaborative Systems. Thessaloniki, Greece.
Nawrocki, J., Jasiński, M., Walter, B., Wojciechowski, A., 2002. Extreme programming modified: embrace requirements engineering practices. In: Proceedings of IEEE Joint International Requirements Engineering Conference. IEEE CS Press, pp. 303–310.
Pendharkar, P.C., Subramanian, G.H., Rodger, J.A., 2005. A probabilistic model for predicting software development effort. IEEE Trans. Softw. Eng. 31 (7), 615–624.
Perkusich, M., Oliveira de Almeida, H., Perkusich, A., 2013. A model to detect problems on scrum-based software development projects. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, pp. 1037–1042.
Radlinski, L., 2010. A survey of Bayesian net models for software development effort prediction. Int. J. Softw. Eng. Comput. 2 (2) ISSN: 2229-7413.
Sarwar, B., Karypis, G., Konstan, J., Riedl, J., 2001. Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web WWW10. Hong Kong, pp. 285–295.
Schulz, T., Radliński, Ł., Gorges, T., Rosenstiel, W., 2010. Defect Cost Flow Model – A Bayesian Network for Predicting Defect Correction Effort. PROMISE 2010, Timisoara, Romania.
Scopus, 2016a. https://www.scopus.com/results/results.uri?sort=plf-f&src=s&sid=639877A6D7D0DCF9E076107E54EA6E4C.wsnAw8kcdt7IPYLO0V48gA%3a210&sot=a&sdt=a&sl=85&s=TITLE-ABS-KEY±%28±bayesian±%29±AND±%28±DOCTYPE±%28±ar±%29±OR±DOCTYPE±%28±cp±%29±OR±DOCTYPE±%28±ip±%29±%29&origin=searchadvanced&editSaveSearch=&txGid=639877A6D7D0DCF9E076107E54EA6E4C.wsnAw8kcdt7IPYLO0V48gA%3a21, (viewed 13 December 2016).
Scopus, 2016b. https://www.scopus.com/results/results.uri?sort=plf-f&src=s&nlo=&nlr=&nls=&sid=639877A6D7D0DCF9E076107E54EA6E4C.wsnAw8kcdt7IPYLO0V48gA%3a330&sot=a&sdt=cl&cluster=scosubjabbr%2c%22COMP%22%2cf%2c%22ENGI%22%2cf%2c%22MATH%22%2cf&sl=85&s=TITLE-ABS-KEY±%28±bayesian±%29±AND±%28±DOCTYPE±%28±ar±%29±OR±DOCTYPE±%28±cp±%29±OR±DOCTYPE±%28±ip±%29±%29&origin=resultslist&zone=leftSideBar&editSaveSearch=&txGid=639877A6D7D0DCF9E076107E54EA6E4C.wsnAw8kcdt7IPYLO0V48gA%3a33, (viewed 13 December 2016).
Si, G., Xu, J., Yang, J., Wen, S., 2014. An evaluation model for dependability of internet-scale software on basis of Bayesian networks and trustworthiness. J. Syst. Softw. 89, 63–75.
Sillitti, A., Succi, G., 2005. Requirements Engineering for Agile Methods Engineering and Managing Software Requirements, Part 2. Springer Berlin Heidelberg, pp. 309–326. doi:10.1007/3-540-28244-0_14.
Standish Group, 2009. Project Smart, http://www.projectsmart.co.uk/the-curious-case-of-the-chaos-report-2009.html.
Tierno, I. A. P., 2013. Assessment of Data-driven Bayesian Networks in Software Effort Prediction, http://hdl.handle.net/10183/71952.
Torkar, R., Awan, N.M., Alvi, A.K., Afzal, W., 2010. Predicting software test effort in iterative development using a dynamic Bayesian network. In: Proceedings of the 21st IEEE International Symposium on Software Reliability Engineering. San Jose, USA.
Usman, M., Mendes, E., Weidt, F., Britto, R., 2014. Effort estimation in agile software development: a systematic literature review. In: Proceedings of the 10th International Conference on Predictive Models in Software Engineering, PROMISE '14. Torino, Italy, pp. 82–91.
Version One, 2007. 2nd Annual Survey "The State of Agile Development care", http://www.versionone.com/pdf/StateOfAgileDevelopmet2_FullDataReport.pdf.
WEKA, 2007a. Mean Absolute Error in Classification, http://weka.8497.n7.nabble.com/Mean-absolute-error-in-classification-td9440.html.
WEKA, 2007b. Root Mean Squared Error Calculation, http://weka.8497.n7.nabble.com/root-mean-squared-error-calculation-td19651.html.
Williams, L., 2010. Agile software development methodologies and practices. Adv. Comput. 80, 1–44.
Wooff, D.A., Goldstein, M., Coolen, F.P.A., 2002. Bayesian graphical models for software testing. IEEE Trans. Softw. Eng. 28 (5), 510–525. doi:10.1109/TSE.2002.1000453.

Srdjana Dragicevic received her M.Sc. degree in electrical engineering from the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. She is currently an Honorary Assistant at the Department of Electronics and Computing of the Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture (FESB), University of Split, Croatia. She is enrolled in the PhD program at FESB, University of Split, Croatia. Her research interests include requirements engineering, decision support, uncertain reasoning, business processes and project management.

Stipe Celar received his B.Sc. degree in electrical engineering from the Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture (FESB), University of Split, Croatia and his Ph.D. degree in technical sciences from TU Wien, Austria. After years of professional work in IT companies and honorary lecturing he is currently an Associate Professor at the Department of Electronics and Computing of FESB, University of Split, Croatia. His research interests include software engineering, software metrics and business information systems.

Mili Turic received his B.Sc. degree in computer science from the Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture (FESB), University of Split, Croatia. He is currently an Honorary Assistant at the Department of Electronics and Computing of FESB, University of Split, Croatia. He is enrolled in the PhD program at FESB, University of Split, Croatia. His research interests include application of software engineering to cost estimation of software projects, and to software project planning in general.