International Journal of Software Engineering and Knowledge Engineering

World Scientific
www.worldscientific.com

REQUIREMENT RISK LEVEL FORECAST USING BAYESIAN NETWORKS CLASSIFIERS

ISABEL MARÍA DEL ÁGUILA

Dpt. Languages and Computation, University of Almería, Carretera de la Playa, Almería, 04120,Spain
imaguila@ual.es

JOSÉ DEL SAGRADO

Dpt. Languages and Computation, University of Almería, Carretera de la Playa, Almería, 04120,Spain
jsagrado@ual.es

Received (15 01 2010)
Revised (15 12 2010)
Accepted (Day Month Year)

Requirement engineering is a key issue in the development of a software project. Like any other development activity is not without risks. This work is about the empirical study of risks of requirements by applying machine learning techniques, specifically Bayesian networks classifiers. We have defined several models to predict the risk level for a given requirement using three dataset that collect metrics taken from the requirement specifications of different projects. The classification accuracy of the Bayesian models obtained is evaluated and compared using several classification performance measures. The results of the experiments show that the Bayesians networks allow obtaining valid predictors. Specifically, a tree augmented network structure shows a competitive experimental performance in all datasets. Besides, the relations established between the variables collected to determine the level of risk in a requirement, match with those set by requirement engineers. We show that Bayesian networks are valid tools for the automation of risks assessment in requirement engineering.

Keywords: Requirement Engineering, Risk assessment, Data Mining, Bayesian Networks Classifiers.

1. Introduction

Software development organizations fail many times to deliverer its products within schedule and budget. Statistical studies, as those conducted by Standish Group, Department of Defense or Software Engineering Institute (SEI) [1, 2], show that, frequently, tasks related to requirements lead software project to the disaster. Problems in requirements have been often cited as one of the highest risks during software life [3]. These problems may be avoided or reduced using systematic and disciplined methods of managing software development risk, specially focused on requirements risk [4, 5].

A risk is an uncertain event that could have a negative outcome; a problem is a risk now materialized. Associated to any risk there are two measures: probability of an unsatisfactory outcome and the loss that will occur if the problem appears [4]. Risk exposure combines these measures. Project managers need to handle risks in a way that

they ensure the success, minimizing potential risk consequences. These managers' tasks are called software risk management.

The goal of risk management in software development is to identify, quantify, plan for, and then react against to potential risks, to keep them from affecting the software project. But as DeMarco [6] confirms, risk management is really project management for adults. This fact combined with that the requirements reside basically in the problem space whereas other software artifacts reside in the solution space [7], make risk management in Requirement Engineering a hard, imprecise and undetermined work.

The most used methods for the identification of the risky software development components are qualitative [8, 9]. Potential risks are assessed by applying subjective risk assessment techniques that are qualitative, human-intensive and error-prone [10]. These assessment tasks should be based on quantitative measures calculated from product attributes. If we are managing risks related to requirements, we would need some metrics about requirements that can give us estimation about the incidence of the potential risks, but nowadays all risk assessment methods currently applied are performed manually.

This work deals with how to apply a specific formalism, Bayesian Networks [11, 12], which originated in artificial intelligence and knowledge engineering fields, as classifiers with the purpose of enhancing activities related to risk management, specifically in requirements activities. This work explores the use of Bayesian networks in requirement engineering, specifically focused on the identification and assessment of risky requirements. The use of Bayesian networks in Software Engineering is not new [13, 14, 15], for example they have been applied in maintenance [15], defect prediction [13] or implementation of a software project [14]. Furthermore, this formalism has also successfully been applied in other issues of Requirement Engineering as in the prediction of the need for a requirements' document review [16] or in the analysis of use cases [17]. Our main purpose is to make a study about the estimation of the risks related to the requirements using automatic learning techniques based on classification trees and Bayesians networks. These techniques will be applied to obtain predictors that indicate the necessity to use risk management techniques to mitigate the exposure to risks. The results obtained by the different predictors will be contrasted against each other. At the time of comparing results, we will use as reference the model based on classification trees. Besides, in the case of predictors based on Bayesian networks, the starting point will be to applied a Naïve Bayes method, that is affected by the hypothesis of independence between predicting variables, and to compare its results with those obtained by other models based on Bayesians networks (with increased tree structure or k-dependencies) that do not make this hypothesis. The various dataset used for the empirical study are freely distributed through the Web pages of the National Aeronautics and Space Administrator (NASA) IV & V facility Metrics Data Program (MDP) repository (http://mdp.ivv.nasa.gov/) and Predictor Models In Software Engineering (PROMISE) projects (http://promisedata.org).

This paper is organized in six sections (excluding this introduction). In Section 2 is described the problem of the risks in Requirement Engineering. The techniques applied to

obtain the predictors are explained in Section 3. The datasets used in this work are included and detailed in Section 4. The experiments designed and the result of the study of the classification performance of Bayesian networks classifiers applied to predict the level of risk for a given requirement, are described in Section 5. The Bayesian classifiers selected on Section 6 gives an explanation of the relationship between the variables used for risk assessment. Due to this reason, this Bayesian network is the generic model that can be used in order to predict the risk level. Finally, in Section 7 the obtained conclusions and the future works that can extend this study are presented.

2. Risks of Requirement Engineering

One of the major problems when developing complex software systems is that of Requirement Engineering. When requirement-related tasks are poorly defined or executed, the software product is typically unsatisfactory [18, 19], and therefore, any improvement in requirements will affect favorably the whole software lifecycle. This is supported by the information collected and analyzed in several reports [19, 20, 1], which point out that about a 70% of software project have troubles and among these an unsuccessful requirements study is the cause of troubles or cancellation in the 50% of the times. Besides if tasks related to requirements are not correctly performed, 35% of all software development projects will not meet the desired and expected quality.

Software requirements express the needs and constraints fixed for a software product that contribute to the solution of some real world problem [21]. Traditionally, obtaining requirements has been considered as a fuzzy step in the software development lifecycle, in which a set of informal ideas must be translated into formal expressions; ambiguity is the rule, not the exception.

Requirements can be considered as the bricks gluing different stages in software project development [22]. So, if we have a risky requirements process, probably we will have a risky project. In order to mitigate the risks, we need to identify and assess risks of requirements, and then, these risks need to be resolved and monitored during a project. Requirement Engineering must be supplemented with Requirement Risk Management in order to avoiding, minimizing, monitoring, mitigating and compensating requirement-based problems [23].

During the last 20 years, many authors have written about risks in software development [4, 5, 3, 24, 25, 26], having as goal the identification of the major risks that can impact software project, the classification of these risks, and the development of more accurate strategies in order to control them. There are many works whose goal is to study and propose mechanisms for risk management in general [8, 4, 9, 27, 28]. Among these works, it is worth to mention the efforts of the Software Engineering Institute (SEI), [8] that offers a wide set of works and technical reports about risks, and defines a model for the management of software development risks, which controls the quality, cost, and schedule of software products. This risk management paradigm follows five cyclic activities: Identification, Analysis, Planning, Track and Control. Identification is about searching for and locating risks that need to be translated into decision-making

information during the analysis. The planning activity is in charge of translating risk information into mitigation actions. Finally, track activity monitors risks and actions and may lead to control actions to correct deviations from the plan.

Risk identification [29, 30] usually is performed by applying taxonomy–based questionnaires that define a framework for identifying the potential risks that can be evolve into a problem in a specific software project. Taxonomy represents an attempt to organize the sources of software development risks around three aspects in software development: Development cycle risks, Development environment risks and Program risks. Development environment and Program risks are not directly related with the software product that is being under development. The development cycle encompasses the tasks associated with software production: requirements gathering, code design, formulation of specifications, project planning, implementation, and testing.

When we face requirements-related risks, we focus on the problem of development cycle risks. The risks included in this category are usually intrinsic risks. That is, risks that can be managed from within the project itself once they have been assessed [30]. Besides, development cycle risks often come from requirements that are difficult or impossible to implement, combined with a lack of an efficient negotiation or incorrect budgets and schedules; from unsuitable analysis of requirements or design specification; or from poor quality design or coding [29]. Thus, many of them come from products or artifacts generated during the project development (i.e. requirements, design specifications).

In the domain of risks related to the stage of gathering and analysis of the requirement, several authors have proposed different kinds of requirements' risks: overlooking a crucial requirement, inadequate customer representation, modeling only functional requirements, not inspecting requirement, attempting to perfect requirement before construction, representing requirement in form of design [3]; developing the wrong functions and properties, developing wrong user interface, gold plating, continuing stream of requirement changes [4]; unrealistic customer expectations, insufficient customer involvement, poor impact analysis, scope creep, defective requirements, new processes and tools [24]. All of them, including [2, 29, 30], focus on the identification of the risks during the Requirements Engineering stage, indicating the origin or the cause of a risk, instead of what are the products with risks. In our case, these are precisely the risky requirements that need to be carefully monitored.

The main approaches to the identification process apply a qualitative and subjective assessment to identify potential risks [29, 30, 31]. A better approach would be that these assessment tasks would be based on quantitative measures calculated from product attributes. If we are managing risks related to requirements, then we will need some metrics about requirements that can give us an estimation about the incidence of the potential risks. The most significant benefits of software metrics is that they provide information to support decision making during software lifecycle [32]. We will have better information for the work of planning and risk control by applying metrics in the process of risk management. The first attempt to use quantitative information in order to

evaluate risks was proposed by Palmer [33]. This author enhances a qualitative risks assessment with a quantitative risk-based metric defining three attributes for each function or requirement to be implemented: volatility, ambiguity and conflict. These values, counts and ratios help user and developer in risk management for the detection of requirements with problems and issues. Other recent works have investigated whether metrics can be used to build predictive models applying textual requirement metrics. Their goal is to identify fault prone software models [34]. Hayes [35] focuses on building a taxonomy of requirement faults working with the data provided by National Aeronautics and Space Administrator (NASA) about some project's metrics. Authors as Feather & Conrford [36] and Wyatt et al. [37] propose predictive models to identify requirement risks, estimating, early in development life cycle, the available metrics about individual requirement.

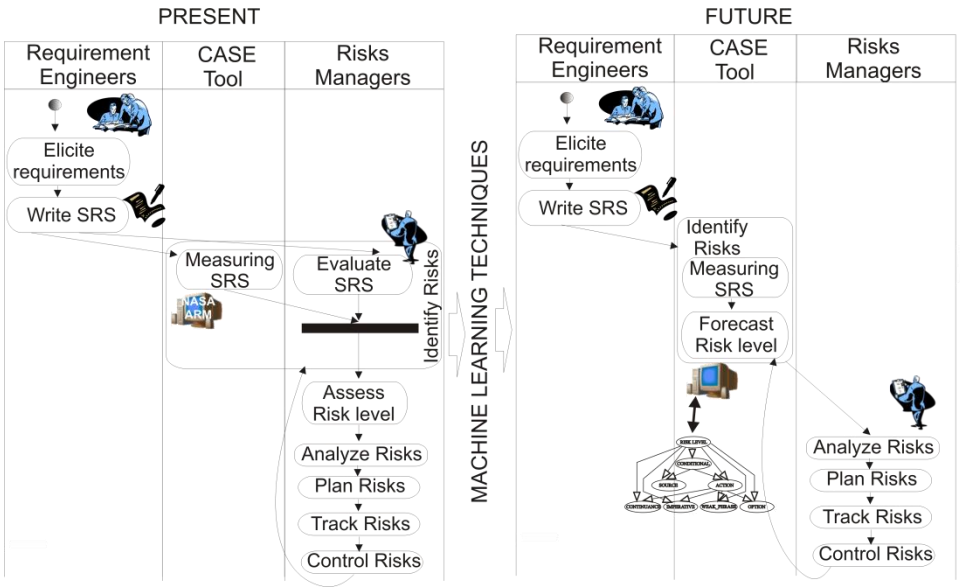


Fig. 1. How to enhance risk management using Bayesian Networks.

With respect to requirements, risk managers have the workload in risk management. The activity diagram under the label PRESENT (see figure 1), shows the workflow that is currently applied. Once requirements have been elicited, and a specification of the requirements (SRS) has been written, requirements are measured by means of software tools (e.g. ARM tool in NASA software IV&V project). These metrics provide information to risks managers in order to identify the risk related to requirements. Then the workflow iterates through the cyclic activities described previously. Our goal is to automate the identification process of risky requirements, as is showed in the activity diagram under the label FUTURE (see figure 1). This could be done using the Bayesian network classifier, if we give it as input the value of one or more metrics for a

requirement obtained from SRS, it returns a risk level assessment for this requirement. In this way the workload of the risk managers is reduced.

Requirements metrics are the basic components that allow us to define the problem of identifying and evaluating risks of an individual requirement as a classification problem. A classification problem tries to separate the objects belonging to a specific domain, into smaller classes, using criteria to determine whether a particular object in the domain belongs or not to a particular class. The information about requirements is that collected on projects such as the NASA Independent Verification & Validation Facility's MDP or PROMISE [38]. In these dataset there are three levels of risk that make up the different classes used for the classification of requirements. One goal is to observe how information (i.e. metrics) collected about a requirement affects to its risk level, and other is to check if the relationships identified in the classification model between requirements' metrics match with those identified naturally by engineers during the process of requirements' risk assessment. We will search for classifiers through the empirical study of such data.

3. Bayesian Networks as Classifiers

The problem of supervised classification is to assign a vector $\boldsymbol{a} = (a_1, \cdots, a_n)$ of attributes or features, one of the *m* classes of the variable C. The true class is denoted by c and takes values in $\{1, 2, \dots, m\}$. There is a cost matrix $cost(r, s)$ with $r, s = 1, \dots m$ which reflects the cost associated with incorrect classifications. In particular $cost(r, s)$ shows the cost of classifying an item of class r in class s. In the case of using the cost function $0/1$, we have

$$cost(r, s) = \begin{cases} 1 & r \neq s \\ 0 & r = s \end{cases} \tag{1}$$

Underlying the observations, we assume the existence of a joint probability distribution which is unknown:

$$p(a_1, \dots, a_n, c) = p(c|a_1, \dots, a_n)p(a_1, \dots, a_n) = p(a_1, \dots, a_n|c)p(c) \tag{2}$$

The aim is to build a classifier that minimizes the total cost of mistakes. This is achieved through the Bayes classifier

$$\gamma(\boldsymbol{a}) = arg \min_k \sum_{c=1}^{m} cos(k, c)\ p(c|a_1, \dots, a_n) \tag{3}$$

In the case that the cost function is $0/1$, the Bayes classifier is equivalent to assigning the instance $\boldsymbol{a} = (a_1, \cdots, a_n)$ the class with the highest posterior probability. That is,

$$\gamma(\boldsymbol{a}) = arg \min_c p(c|a_1, \dots, a_n) \tag{4}$$

In practice, the joint distribution function $p(a_1, \cdots, a_n, c)$ is unknown, and can be estimated from a simple random sample $\{(\boldsymbol{a}^{(1)}, c^{(1)}), \dots, (\boldsymbol{a}^{(N)}, c^{(N)})\}$ that is supposed extracted from the joint distribution function.

3.1. *Naïve Bayes*

Under this name is known the classification paradigm that uses Bayes' theorem in conjunction with the hypothesis of conditional independence of the predictors variables given the class [39, 40]

The naïve Bayes paradigm is based on two premises established on the predictors (findings) and the variable to predict (diagnosis). These premises are:

- Diagnoses are mutually exclusive. That is, the variable to predict C takes one of its m possible values.
- Findings are conditionally independent given the diagnosis. That is, if you know the value of the diagnosis variable, knowledge of any of the findings is irrelevant to the other findings.

$$p(A_1 = a_1, \dots, A_n = a_n | C = c) = \prod_{i=1}^{n} p(A_i = a_i | C = c) \tag{5}$$

Therefore, in the naïve Bayes paradigm, finding most likely diagnosis, c^*, once known symptoms $(a_1, \dots a_n)$ for a particular patient, reduces to find

$$c^* = arg \max_c p(c = c) \prod_{i=1}^{n} p(A_i = a_i | C = c) \tag{6}$$

Trying to overcome the strong constraints underlying in the naïve Bayes paradigm, other paradigms have been developed allowing us to express dependency relationships between the predictors.

3.2. *Tree Augmented Naïve Bayes*

In order to obtain a naïve Bayes classifier with a tree enhanced structure, we start with a tree structure with the predictors variables, to later connect the class variable with each of the predictor variables.

Friedman et al. [41] present an algorithm called tree augmented network (TAN), which is basically an adaptation of the Chow-Liu algorithm [42]. It takes into account the amount of mutual information conditional on the class variable. The amount of mutual information between discrete variables A, B conditional on the variable C is defined as

$$I(A, B|C) = \sum_{i=1}^{t} \sum_{j=1}^{u} \sum_{k=1}^{v} p(a_i, b_j, c_k) \, log \frac{p(a_i, b_j | c_k)}{p(a_i |, c_k) p(b_j | c_k)} \tag{7}$$

If the data has been generated by a tree-shaped structure, the TAN algorithm is asymptotically correct, in the sense that if the sample is large enough, it will recall the structure that generated the file of cases.

3.3. *K-Dependent Bayesian Classifiers*

Sahami [43] proposed an algorithm called k dependence Bayesian classifier (KDB), which enables traverse the wide spectrum of dependences available between the naïve Bayes model and the model corresponding to a complete Bayesian network. The algorithm is based on the concept of k-dependent Bayesian classifier, which contains the structure of the naïve Bayes classifier and allows each predictor have a maximum of k parents variables without counting the class variable.

Thus, the naïve Bayes model corresponds to a 0-dependent Bayesian classifier, a TAN model would be a 1-dependent Bayesian classifier and a complete Bayesian classifier (the structure does not reflect any independence) would correspond to a (n-1)-dependent Bayesian classifier.

The basic idea of the algorithm is to generalize the algorithm proposed by Friedman et al [41] allowing each variable to have a number of parents, without counting the class variable C, bounded by k.

4. NASA IV & V requirement metrics

The NASA IV & V facility Metrics Data Program (MDP) repository provides access to software metrics and the associated error data at the function/method level for NASA software development projects. The repository is a database that stores problem data, product data and metrics data. These data provides the opportunity to investigate the relationship of metrics or combinations of metrics to the software.

Measures can help in the understanding of software and the Software Engineering processes in order to derive models of those processes and examine relationships among the process parameters. The software measurement guidebook [44] provides three key reasons for software measurement:

1. Understand and model software engineering processes and products
2. Aid in the management of software projects
3. Guide improvements in software engineering processes

Table 1. NASA MDP Requirement metrics (from http://mdp.ivv.nasa.gov/).

Measure	Description	Observation
Identifier	Unique requirement Identifier	
Action	Represents the number of actions the requirement needs to be capable of performing	Manual assignment (MA)
Conditional	Represents whether the requirement will be addressing more than one condition. This indicates a higher level of complexity in dealing with multiple conditions within the requirement (i.e., If, when, in the event of).	MA
Continuance	Phrases such as "the following:" that follow an imperative and precede the definition of lower level requirement specification. The extent that continuances are used is an indication that requirements have been organized and structured. These characteristics contribute to the tractability and maintenance of the subject requirement specification. However, extensive use of continuances indicate multiple, complex requirements that may not be adequately factored into development resource and schedule estimates	Automated Requirement Measurement software tool (ARM)
Imperative	Those words and phrases that command that something must be provided. "Shall" normally dictates the provision of a functional capability. "Must" or "must not", normally establish performance requirements or constraints. "Will" normally indicates that something will be provided from outside the capability being specified. The ARM report lists the imperatives and their associated counts in descending order of forcefulness. An explicit specification will have most of its counts high in the report IMPERATIVE list (i.e. shall, must, required)	ARM
Incomplete	Phrases such as "TBD" or "TBR". They are used when a requirement has yet to be determined. These are considered critical to requirements documents and need to be corrected as soon as possible. They can cause unexpected delays and high costs.	ARM
Option	Those words that give the developer latitude in the implementation of the specification that contains them. This type of statement loosens the specification, reduces the acquirer's control over the final product, and establishes a basis for possible cost and schedule risks	ARM
Risk Level	A calculated risk level metric based on weighted averages from metrics collected for each requirement. Level 1: Indicates a non-complex straight forward requirement containing one imperative, single action and single source. Level 2: Indicates a requirement containing multiple imperatives, more than one action and/or more than one source. Level 3: Indicates a requirement containing conditionals and more than one action and/or source	MA
Source	Represents the number of sources the requirement will interface with or receive data from.	MA
Weak Phrase	Clauses that are apt to cause uncertainty and leave room for multiple interpretations. Use of phrases such as "adequate" and "as appropriate" indicate that what is required is either defined elsewhere or worst, the requirement is open to subjective interpretation. Phrases such as "but not limited to" and "as a minimum" provide the basis for expanding requirements that have been identified or adding future requirements. Weak Phrase total is indication of the extent that the specification is ambiguous and incomplete.	ARM

We use different datasets from NASA MDP repository. Each one contains many metrics, which describe product´s size, complexity and some structural properties. Only 3 of the 13 project included in the NASA MDP offer requirement metrics. As Table 1 shows the assignment of values of some of these metrics is performed manually (MA) whereas the values assignment of other metrics is performed making use of natural language processing (NLP) techniques. NLP methods are also used in commercial tools for the analysis of requirement's quality, such as the QualityAnalyzer for DOORS and IRqA (http://www.reusecompany.com).

The NASA projects used are: CM1 project is a NASA spacecraft instrument, JM1 is a real-time prediction ground system, PC1 is a flight software for earth orbiting satellite. All are writing in C and the numbers of modules are, respectively, 498, 10885 and 1109. Table 1 shows the set of 10 attributes used to describe requirements in NASA MDP. Table 2 shows the analysis of requirement measures contained in each dataset and the number of instances for each of the three requirements' risk levels considered.

Table 2. Requirement Measures Analysis

Measure	CM1 (160 instances)				JM1 (74 instances)				PC1 (320 instances)			
	Min	Max	Mean	Stdev	Min	Max	Mean	Stdev	Min	Max	Mean	Stdev
Action	1	5	1.463	0.768	0	5	1.514	0.996	1	6	1.659	0.976
Conditional	0	1	0.144	0.352	0	1	0.284	0.454	0	1	0.234	0.424
Continuance	0	3	0.425	0.61	0	4	0.595	0.978	0	5	0.666	0.894
Imperative	1	4	1.238	0.599	0	5	1.392	1.031	0	5	0.816	0.885
Incomplete	0	0	0	0	0	0	0	0	0	0	0	0
Option	0	1	0.025	0.157	0	1	0.081	0.275	0	3	0.019	0.193
Source	1	5	1.55	0.896	0	6	1.797	1.238	1	23	2.072	2.034
Weak phrase	0	1	0.125	0.332	0	1	0.108	0.313	0	2	0.013	0.137
	Level 1	Level 2	Level 3		Level 1	Level 2	Level 3		Level 1	Level 2	Level 3	
Risk Level	68	58	34		24	23	27		168	104	48	

Risk level is assigned manually to each requirement during requirement analysis through a revision of the metrics collected for each requirement. Thus, Risk level 3 corresponds to requirements having conditional and/or incomplete sentences together with multiple imperatives, actions or sources. These requirements are the most complex for implementation and require additional testing. Risk level 2 corresponds to requirements that can contain imperatives, weak phrases, options and/or continuances. They are moderately complex requirements that do not contain conditional or incomplete sentences. Risk level 1 is associated to requirements with weak phrases, options and/or imperatives and that are defined in a clear and concise way.

5. Bayesian Network Classifiers applied to Risk of RE

Once the attributes (i.e. action, conditional, continuance, imperative, option, source, weak phrase) and class (i.e. requirement risk level) have been identified in the previous section, we are going to apply Bayesian networks in order to classify the risk associated to a given requirement. Notice that attribute "incomplete" has assigned a zero value in all the cases of the three datasets been considered. This is the reason why we have not taken it into account as an attribute in the classification problem.

In order to develop the experimental study we have applied the same schema in the different datasets:

1. Learn using stratified tenfold cross-validation a model for each of the different types of Bayesian networks classifiers proposed and a classification tree
2. Obtain the classification results of each classifier as a contingency table.
3. Evaluate and compare the classification accuracy achieved by the different classifiers.

First, we have learnt using stratified tenfold cross-validation a model each of the different types of Bayesian networks classifiers proposed and a classification tree. That is, the original dataset is divided into ten parts, each of which preserves the properties of the original sample, using nine for learning and one for testing. The learning-testing process is repeated ten times, one for each partition. The classification tree will be used as reference at the time of comparing and evaluating results, because it is a commonly used technique in data mining. Second, we have obtained the classification results of each classifier as a contingency table with a row and a column for each class. Each element of the contingence table shows the number of instances for which the actual class is the row and the predicted class is the column. In our problem, the predicted variable is the risk level for a given requirement, so we have a 3x3 contingency table as is depicted in Table 3 collecting prediction outcomes for each classifier.

Table 3. Contingency table for requirements' level of risk prediction.

		Predicted class		
		Risk Level 1	Risk Level 2	Risk Level 3
Actual class	Risk Level 1	a	b	c
	Risk Level 2	d	e	i
	Risk Level 3	g	h	j

Third, a comparison between the different models applied to predict the risk level for a given requirement, is performed evaluating and comparing the classification accuracy achieved by the different classifiers using several measures [45] (the first five can be calculated directly from the contingency table):

- *Percentage of correctly classified instances*, for example for the contingency matrix shown in Table 3 this value will be computed as. *(a+e+i)/(a+b+c+d+e+f+g+h+i)*
- *True positive* (TP) rate for each class is the proportion of instances that were correctly classified, among all instances which truly belongs to the class. For example, the TP for "Risk level 1" class is *a/(a+b+c)*. It is equivalent to *recall*, i.e. the portion of the class that was captured, and give us a measure of completeness.
- *False positive* (FP) rate for each class is the proportion of instances that were incorrectly classified in the class, among all the instances which do not belong to the class. For example, the FP for "Risk level 1" class is *(d+g)/(d+e+f+g+h+i)*.
- *Precision* is the proportion of instances which truly belong to a given class among all those which were classified in the class, i.e. *a/(a+d+g)* for "risk level 1". It is a measure of exactness or fidelity that tells us the probability of a correct classification of the risk level of a given requirement.
- *F-measure* is a measure of accuracy and is defined as the harmonic mean of precision and recall: *F=2·(precision·recall)/(precision+recall)*.
- *Relative Operating Characteristic* (ROC) curve and the *Area Under ROC Curve* (AUC). The ROC curve provides a graphical representation of the classification performance by means of depicting the TP rate as function of the FP rate across all the possible experimental settings. A ROC curve allows visual examination of the tradeoff between the ability of a classifier to correctly detect risky requirements (*recall*) and the number of requirements whose risk level is incorrectly classified (*false positive*). The classification accuracy can be measured by the AUC (an area of 1 represents a perfect test whilst an area of 0.5 corresponds to a worthless test). As Fawcett [46], points out, this measure can be interpreted as the probability that when we randomly pick one positive and one negative example, the classifier will assign a higher score to the positive example than to the negative.

Table 4. Percentage of Correctly Classified Instances

Dataset	J48	Naïve	TAN	KNN	BN
CM1	91,88	**92,50**	**92,50**	91,25	91,25
JM1	79,73	75,68	**85,14**	83,78	83,78
PC1	**91,88**	85,31	86,25	86,25	87,19

The values obtained for these measures in the different datasets are shown in Tables 4, 5, 6, 7.

Table 4 shows the percentage of correctly classified instances obtained by each classifier in each dataset. The best percentages are highlighted in boldface suggesting that TAN is the most appropriate schema for a Bayesian classifier that predicts the risk level of a requirement.

From the viewpoint of a software project, risk levels 2 and 3 are the most important, because requirements within these levels are those that need to be monitored throughout the project. Requirements with a high level of risk can affect the project in a way that it cannot meet quality, deadlines or, even worst, that the project can be canceled. An error in classifying the risk level associated with one of these requirements affect the entire subsequent development of the project.

On the other hand, if we assign a risk level 2 or 3 to a requirement which has a risk level of 1, we increase the cost of the software project, because we are expending resources and taking control actions to manage a requirement which need not be so closely controlled.

A requirement with risk level 1 is a requirement that is clearly defined and not too complex, which is unlikely to present problems in its implementation on the delivered software product. These requirements have less influence on the project risk.

Table 5. Performance Results for CM1

		TP Rate	FP Rate	Precision	F-measure	AUC
Risk Level 1	J48	**1,000**	0,076	0,907	0,951	0,986
	Naïve	**1,000**	0,033	0,958	0,978	**1,000**
	TAN	**1,000**	0,033	0,958	0,978	0,998
	KNN	**1,000**	**0,022**	**0,971**	**0,986**	0,997
	BN	**1,000**	**0,022**	**0,971**	**0,986**	**1,000**
Risk Level 2	J48	0,828	**0,029**	**0,941**	0,881	0,929
	Naïve	0,897	0,059	0,897	0,897	0,960
	TAN	**0,914**	0,059	0,898	**0,906**	0,962
	KNN	0,897	0,078	0,867	0,881	0,966
	BN	0,879	0,069	0,879	0,879	**0,972**
Risk Level 3	J48	**0,912**	**0,024**	**0,912**	**0,912**	0,956
	Naïve	0,824	**0,024**	0,903	0,862	0,976
	TAN	0,794	**0,024**	0,900	0,844	**0,979**
	KNN	0,765	0,032	0,867	0,813	0,978
	BN	0,794	0,040	0,844	0,818	0,977

Table 6. Performance Results for JM1

		TP Rate	FP Rate	Precision	F-measure	AUC
	J48	**0,958**	**0,040**	**0,920**	**0,939**	0,958
	Naïve	**0,958**	0,080	0,852	0,902	0,962
Risk Level 1	TAN	**0,958**	**0,040**	**0,920**	**0,939**	**0,966**
	KNN	**0,958**	**0,040**	**0,920**	**0,939**	0,963
	BN	**0,958**	**0,040**	**0,920**	**0,939**	0,963
	J48	0,652	0,137	0,682	0,667	0,851
	Naïve	0,609	0,176	0,609	0,609	0,789
Risk Level 2	*TAN*	**0,739**	0,098	0,773	**0,756**	**0,890**
	KNN	0,652	**0,078**	**0,789**	0,714	0,883
	BN	0,652	**0,078**	**0,789**	0,714	0,878
	J48	0,778	0,128	0,778	0,778	0,863
	Naïve	0,704	0,106	0,792	0,745	0,915
Risk Level 3	*TAN*	0,852	**0,085**	**0,852**	**0,852**	**0,968**
	KNN	**0,889**	0,128	0,800	0,842	**0,968**
	BN	**0,889**	0,128	0,800	0,842	0,967

Table 7. Performance Results for PC1

		TP Rate	FP Rate	Precision	F-measure	AUC
	J48	**0,970**	**0,053**	**0,953**	**0,962**	**0,979**
	Naïve	0,899	0,099	0,910	0,904	0,973
Risk Level 1	TAN	0,917	0,118	0,895	0,906	0,974
	KNN	0,923	0,118	0,896	0,909	0,973
	BN	0,917	0,125	0,890	0,903	0,970
	J48	**0,904**	**0,074**	**0,855**	**0,879**	0,939
	Naïve	0,788	0,111	0,774	0,781	0,935
Risk Level 2	*TAN*	0,788	0,097	0,796	0,792	**0,944**
	KNN	0,788	0,097	0,796	0,792	0,941
	BN	0,817	0,097	0,802	0,810	0,939
	J48	0,771	0,007	0,949	0,851	0,927
	Naïve	**0,833**	0,029	0,833	0,833	**0,977**
Risk Level 3	*TAN*	**0,833**	0,018	0,889	0,860	0,976
	KNN	0,813	0,018	0,886	0,848	0,974
	BN	0,833	**0,004**	**0,976**	**0,899**	**0,977**

Tables 5, 6, 7 show the performance measures associated to each dataset. These measures are analyzed taking into account the nature of the risks in the requirement domain. Thus, in order to select the most appropriate structure for the Bayesian classifier, we consider the two structures that have achieved best values on the performance measures associated to risk levels 3 and 2. The values highlighted in boldface in these tables correspond to the maximum value reached, except in the case of FP-rate, which is the lowest one. Therefore, the selected structures are Naïve y TAN in the case of the dataset CM1, TAN and KNN for JM1, and TAN and NB in the case of PC1.

In order to make more comprehensive the comparative, the Figures 2, 3, 4 depict the ROC curves associated to the above selected Bayesian classifiers and to the classification tree which we take as a reference model.

We fix our attention on two regions when examining ROC curves. First region is unfavorable for the cost and covers the initial part of the ROC curve with low values for the rates of false positives and true positives. The second region is unfavorable for risk and covers the final part of the ROC curve with high values for both rates. As was previously pointed out, in the problem at hand is more important to identify the risk level of a requirement that the cost of reviewing its level of risk. Thereto, we focus our attention on the risk adverse region, preferring classifiers whose performance is higher in the unfavorable risk area for the different risk levels.

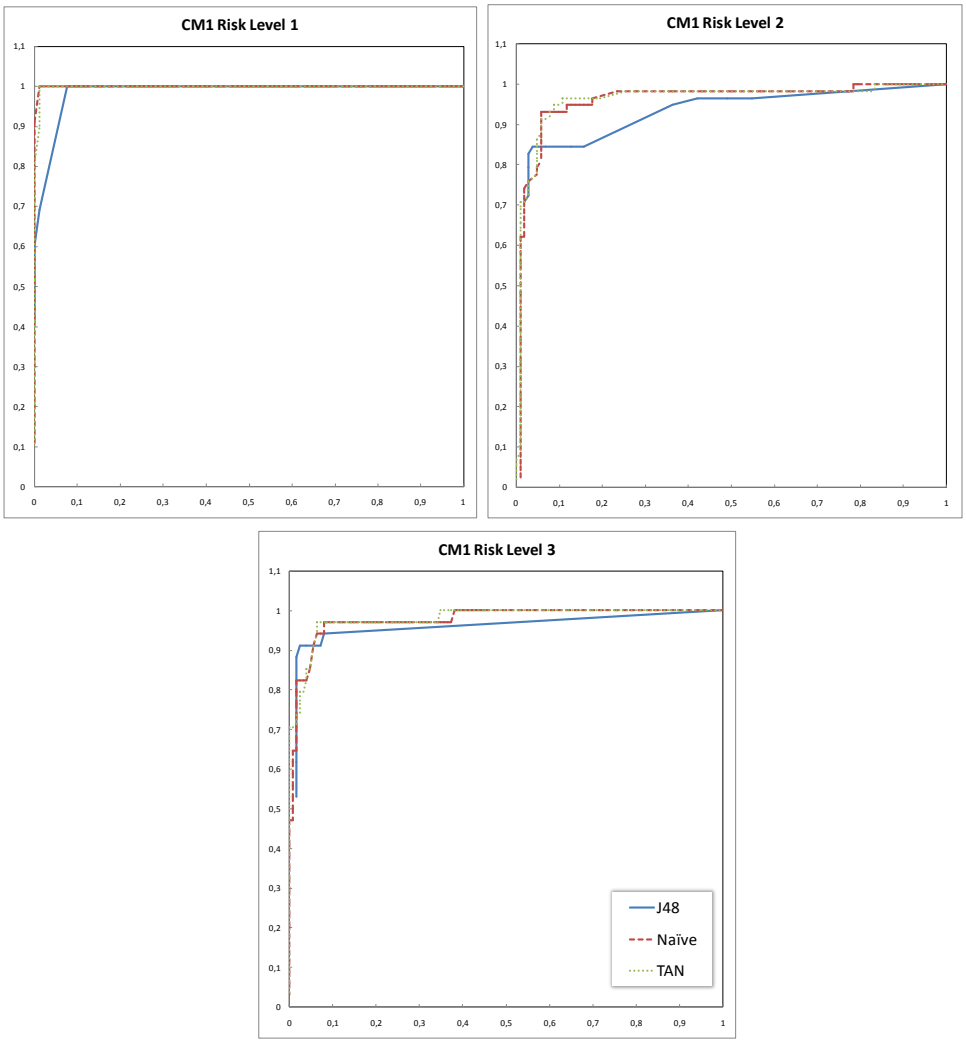


Fig. 2. ROC curve of CM1 dataset.

Figure 2 shows the equality on performance achieved for the CM1 data set between Naïve and TAN, as is reflected by the data in Table 5. The same fact is shown in Figures 3 and 4, between TAN and KNN, and BN y TAN, for the datasets JM1 and PC1 respectively. It is worth to notice that in all cases TAN shows a competitive behavior, i.e. it is the second best in CM1, the best in JM1 and the third best in PC1.

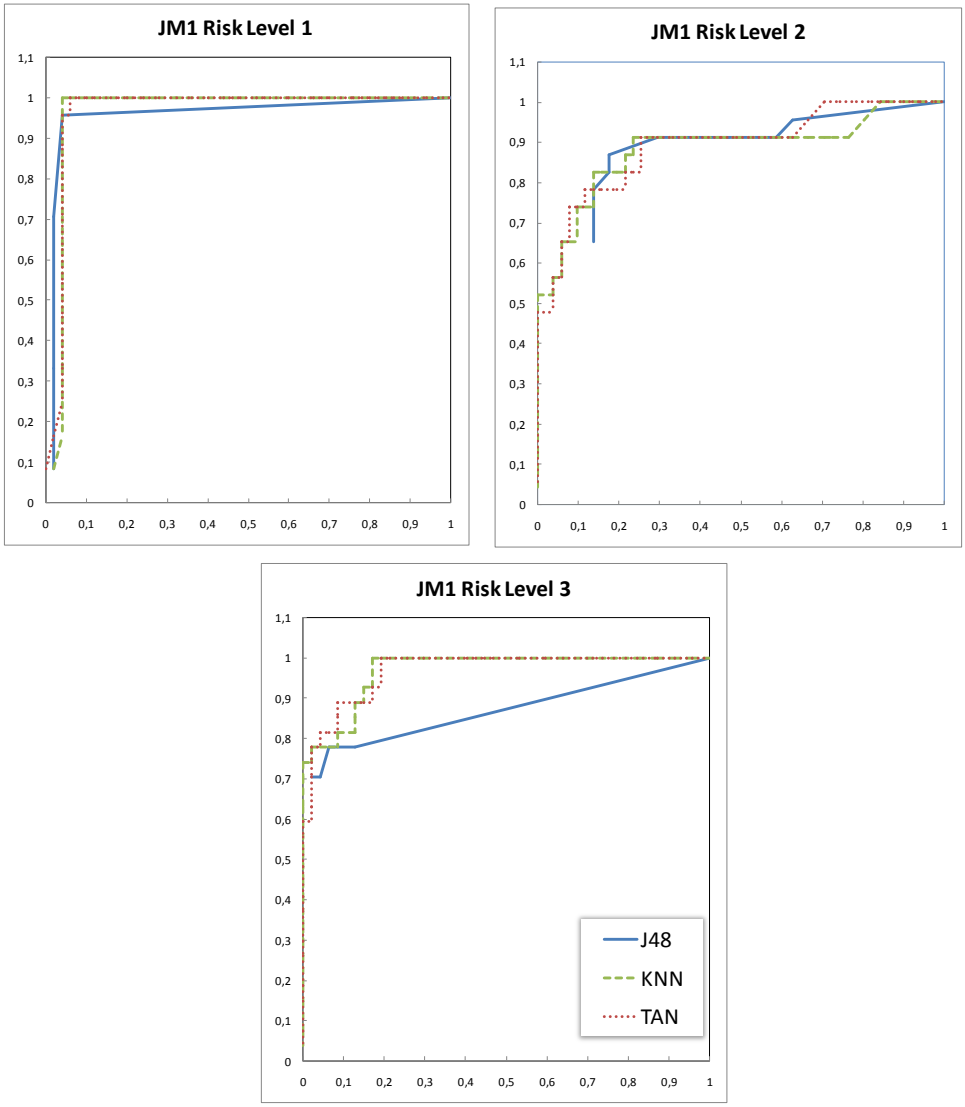


Fig. 3. ROC curve of JM1 dataset.

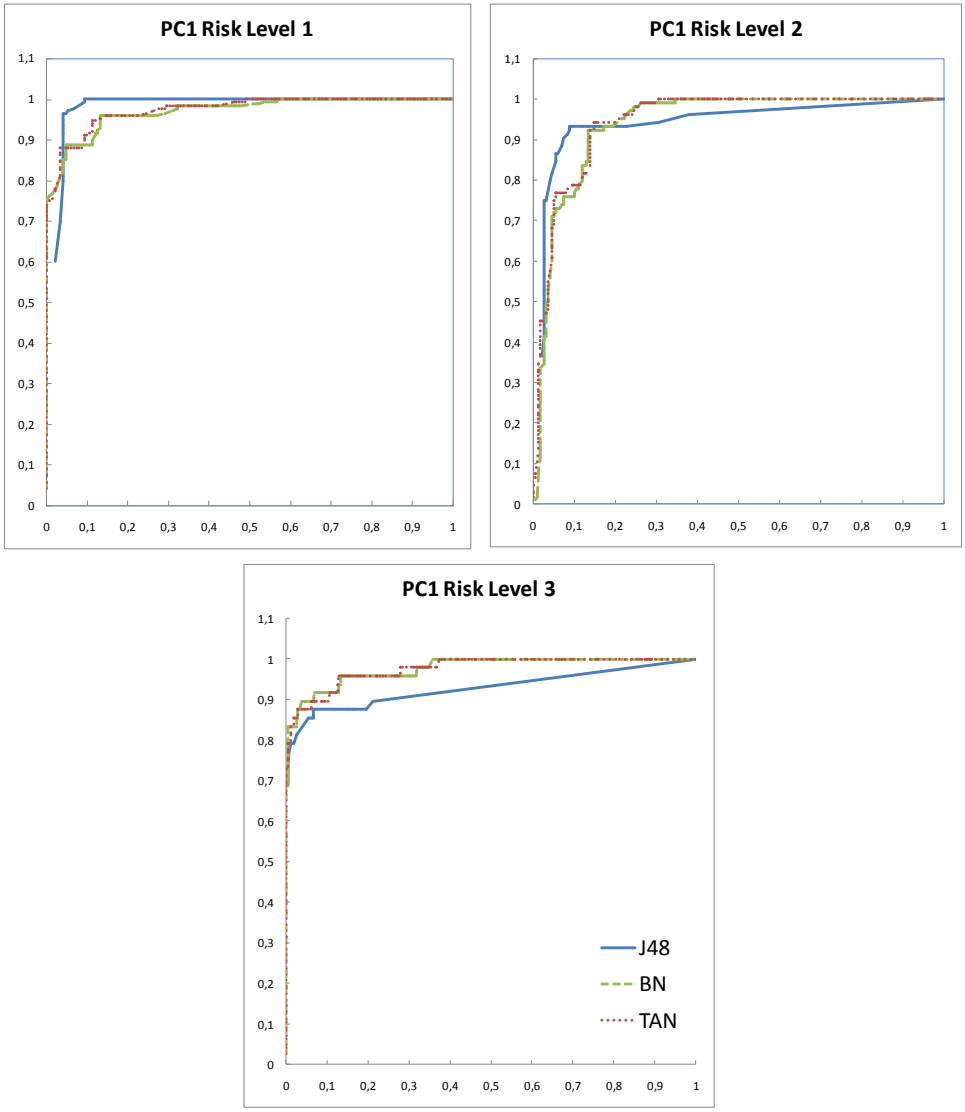
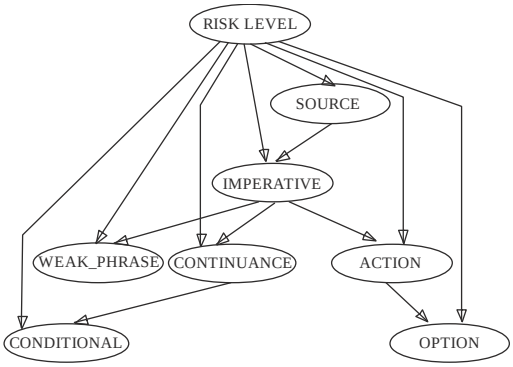


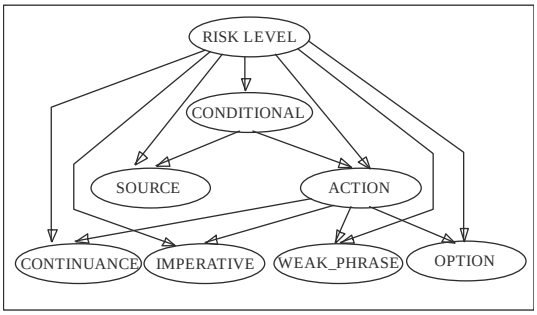
Fig. 4. ROC curve of PC1 dataset.

6. A Tree Augmented Network for classifying the risk level of requirements

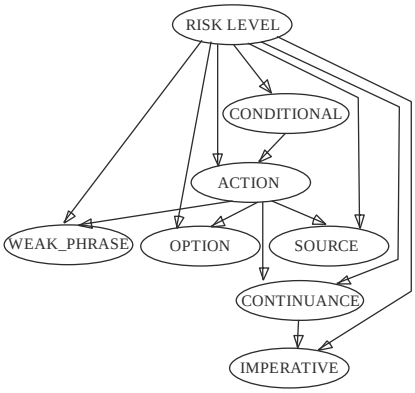
The classification performance exhibited by the TAN classifier leads us to consider whether we can find a Bayesian classifier that works on all data sets and, at the same time, explain the relationship between the variables used for the allocation of risk to requirements that are proposed in the MDP of NASA (http://mdp.ivv.nasa.gov/).



(a) TAN structure for CM1



(a) TAN structure for JM1 (RLRTAN)



(a) TAN structure for PC1

Fig. 5. Different TAN structures learned from dataset.

Figure 5 shows the three TAN structures learned. Of those three structures, that obtained on JM1 (Fig. 5.b) represents relationships between requirement metrics (the direct relationships are represented as arcs in the structure) that can be explained in a natural way from a risk manager point of view. The fact that a requirement has a high level of complexity (conditional), influences directly on the number of sources that it has to interface with (source) and on the number of actions that it has to be capable of performing (actions), in an increasing way. Thus, if a requirement has to be able to perform a high number of actions, there is a higher chance that its explicit specification will have a greater number of imperative terms telling that something must be provided (imperative). When the number of actions increases, also increase the need for a greater organization and a better structure in the requirement (continuance). Moreover, the increase of actions reduces the acquirer's control over the final product. In other words the value of the metric "option" increases. Finally, the increase in the number of actions has also a direct impact on the extent that the specification is ambiguous and incomplete (weak phrase). This is the reason why we have decided to select it and call it Risk Level of Requirements TAN (RLRTAN).

In order to check the validity of these claims, we have kept the RLRTAN structure on CM1 and PC1 sets to estimate the model parameters and evaluate the performance of the classifier. The results are shown in Table 8 and from them it follows that RLRTAN improves the percentage of correctly classified instances in 1.25% for CM1 and 0.63% for PC1. This improvement is also seen in the other measures used (compare Tables 5 and 7 with Table 8). To ensure that this structure is the most suitable from among those used in this study, we have check experimentally that the KNN structure learned from JM1 and the Bayesian network learned from PC1, when kept fixed and its parameters are learned in the other data sets, do not provide better results than the models previously considered, but make them worse.

Table 8. Performance Results for RLRTAN.

Data set	% Correct Classification	Risk Level	TP Rate	FP Rate	Precision	F-measure	AUC
CM1	93,75	1	1,000	0,011	0,986	0,993	1,000
		2	0,931	0,059	0,900	0,915	0,971
		3	0,824	0,024	0,903	0,862	0,979
PC1	86,88	1	0,923	0,105	0,906	0,914	0,972
		2	0,808	0,102	0,792	0,800	0,938
		3	0,813	0,015	0,907	0,857	0,965

Now once a knowledge model, RLRTAN, has been built, then it can be used to assess the risk level for a requirement. The Bayesian network receives as input the values of the metrics associated to a given requirement and through an inference process computes its risk level.

7. Conclusions and Future works

The risks assessment problem in Requirement Engineering traditionally is solved based on the skills, capabilities and experience of developers. In order to facilitate this assessment, requirements metrics must be the basic components that allow us to define the problem of identification and evaluation of risks of an individual requirement. If we focus on the identification of the risky requirements that need to be carefully monitored, then this problem can be redefined as a classification problem.

In this paper we have developed an empirical study using several datasets that collect metrics taken from the requirement specifications in three different NASA projects, developed for a spacecraft instrument, a real time prediction ground system and a flight software for orbiting satellite. In these datasets the risk level of a given requirement was fixed manually, based on a set of requirement metrics, between three classes that correspond to the values high, medium, and low that can be assess to risk level.

In our experimental study, we have applied the same schema in the different datasets. First, we have learnt using stratified tenfold cross-validation a model each of the different types of Bayesian networks classifiers proposed and a classification tree. Then, we have obtained the classification results and compute several performance measures (i.e. FP-rate, TP-rate, F-measure, AUC). We have selected the most appropriate structure for the Bayesian classifier, considering the two structures that have achieved best classification performance values taking into account that it is more important to identify the risk level of a requirement that the cost of reviewing its level of risk.

The classification performance exhibited by the TAN classifier leads us to consider whether we can find a Bayesian classifier that works on all datasets and, at the same time, explain the relationship between the variables used for the allocation of risk to requirements. In consequence, we have obtained a TAN structure (RLTAN) whose set of relationships reflects the way in which requirements' attributes characterize risks.

In addition to these conclusions, we can say that the available data are not as extensive as it would be necessary. Nonetheless, they are related with those Requirement Engineering approaches in which the requirement specification is a document. The actual approaches in requirement development usually are assisted by requirement management tools, making the requirement specification something more than just a document. These tools offer the possibility of defining some other more useful metrics, i.e. the number of changes that a requirement has experienced; the relative growth of a requirement; the number of stakeholders that has proposed it; if the requirement has been reused from other projects; etc. As future work, we plan to extend this study using new metrics and to incorporate the classifiers obtained as an aid facility in requirement management tools in order to obtain automatic risk predictions.

Acknowledgments

This work was supported by the Spanish Ministry of Education and Science under project TIN2010-20900-C04-02 and by the Junta of Andalucía under project TEP-06174.

References

[1] Standish Group, *CHAOS Summary 2009*, (Standish group, Boston, MA, USA, 2009).
[2] T.R. Leishman, D.A. Cook, Requirements Risks Can Drown Software Projects CrossTalk – *Journal of Defense Software Engineering* (April 2002). http://www.stsc.hill.af.mil/crosstalk/2002/04/leishman.html
[3] B. Lawrence, K. Wiegers and C. Ebert, The Top Risks of Requirements Engineering, *IEEE Software* **18** (2001) 62-63.
[4] B. Boehm, Software risk management: principles and practices, *IEEE Software* **8**(1) (1991) 32-41.
[5] R. Fairley, Risk Management for Software Projects, *IEEE Software* **11**(3) (1994) 57-67.
[6] T. DeMarco and T. Lister, Risk Management during Requirements, *IEEE Software* **20**(5) (2005) 99-101.
[7] B. H. C. Cheng and J.-M. Atlee, Research Directions in Requirements Engineering, in *Proc of Future of Software Engineering (FOSE 2007)*, Minneapolis, MN, USA, 2007, pp 285-303.
[8] A.J. Dorofee, *Continuous Risk Management Guidebook First Edition* (Carnegie Melon University, Pittsburgh, PA, USA, 1996).
[9] R.C. Williams, G.J. Pandelios and S.G. Behrens, Software Risk Evaluation (SRE) Method Description (Version 2.0), CMU-Technical Report, SEI-99-TR-029, (Ed.: Software Engineering Institute, Carnegie Mellon University, 1999)
[10] N.F. Schneidewind, Predicting risk as a function of risk factors, *Innovations in Systems and Software Engineering* **1**(1) (2005) 63-70.
[11] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference* (Morgan Kaufman, San Mateo, CA, 1988).
[12] F.V. Jensen, *Bayesian Networks and decision graphs* (Springer-Verlag, 2001).
[13] N. Fenton, M. Neil, W. Marsh, P. Hearty, D. Marquez, P. Krause and R. Mishra, Predicting software defects in varying development lifecycles using Bayesian nets, *Information and Software Technology* **49**(1) (2007) 32-43.
[14] E.J. Lauria and P.J. Duchessi, A Bayesian Belief Network for IT implementation decision support, *Decision Support Systems* **42**(3) (2006) 1573-1588.
[15] A.C. de Melo and A.J. Sanchez, Software maintenance project delays prediction using Bayesian Networks, *Expert Systems with Applications* **34**(2) (2008) 908-919.
[16] J. del Sagrado, and I.M. del Águila. A Bayesian Network for Predicting the Need for a Requirements Review. In *Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects*, (Ed.: Farid Meziane and Sunil Vadera, IGI Global, Hershey PA, USA , 2009), pp. 106-128.
[17] Ganesh Pai, Joanne Bechta-Dugan, Khalid Lateef, "Bayesian Networks applied to Software IV&V," Software Engineering Workshop, Annual IEEE/NASA Goddard, pp. 293-304, 29th Annual IEEE/NASA Software Engineering Workshop, 2005
[18] I. Sommerville, *Software Engineering: (Update) (8th Edition)* (Addison Wesley, Essex, England, 2006).
[19] R.L. Glass, *Facts and Fallacies of Software Engineering* (Addison Wesley, Boston, MA, 2002).
[20] Standish Group, *Chaos Chronicles v3.0.* (Standish group, Boston, MA, USA, 1999).

[21] G. Kotonya and I. Sommerville, *Requirements Engineering: Processes and Techniques* (Wiley, New York, NY, 1998).
[22] C. Ebert, Understanding the product life cycle: four key requirements engineering techniques, *IEEE Software* **23**(3) (2006) 19-25.
[23] D. Gelperine, Identifying and Controlling Requirements Risk (v40 1.2M) Technical Report (Ed: Clearspecs, Posted: 6/15/2009), http://clearspecs.com/joomla15/.
[24] Ellen Gottesdiener, How Agile Practices Reduce Requirements Risk, *Better Software*, **Jul/Ago** (2009) 14-15.
[25] C. Chittister and Y. Haimes, Fellow, Risk Associated with Software Development, *IEEE Transactions on Systems, Man and Cybernetics* **23**(3) (1993) 14-15.
[26] M. Keil, P.E. Cule, K. Lyytinen and R.C. Schmidt, A Framework for Identifying Software Project Risks, *Communication of the ACM* **41**(11) (1998) 76-83.
[27] G. Stoneburner, A.Goguen, and A. Feringa, Risk Management Guide for Information Technology Systems, Report Nº SP 800-30 (Ed.: National Institute of Standards and Technology, 2002).
[28] F. Brown, K. Canal, R. Cullen, M. Elliott, J. Faulkner, M. Lackner, C. Owens, D. Peercy, G.D. Reisz, P. Tempel and P. Trellue, Software Risk Management A Practical Guide, Report Nº SQAS21.01.00 (Ed.: Software Quality Assurance Subcommittee, Department of Energy, 1999).
[29] M.J. Carr, S.L. Konda, I. Monarch, F.C. Ulrich and C. F. Walker, Taxonomy-Based Risk Identification, CMU-Technical Report SEI-93-TR-6 ESC-TR-93-183, (Ed.: Software Engineering Institute, Carnegie Mellon University, 1993).
[30] R.P. Kendall, D.E. Post, J.C. Carver, D.B. Henderson and D.A. Fisher, A Proposed, Taxonomy for Software Development Risks for High-Performance Computing (HPC) Scientific/Engineering Applications, CMU-Technical Notes SEI-2006-TN-039, (Ed.: Software Engineering Institute, Carnegie Mellon University, 2007).
[31] H. Barki, S. Rivard and J. Talbot Toward an assessment of software development risk, *Journal of Management Information Systems* **10**(2) (1993) 203-225.
[32] N.E. Fenton and M.Neil, Software metrics: successes, failures and new directions, *Journal of Systems and Software* **47**(2-3) (1999) 149-157.
[33] J.D.Palmer, and R.P. Evans, Software risk management: requirements-based risk metrics, In *Proc of International Conference on Systems, Man, and Cybernetics, 1994. 'Humans, Information and Technology'*, vol.1 San Antonio, Texas, USA, 1994, pp. 836-841
[34] Y. Jiang, B. Cukic and Y. Ma: Techniques for evaluating fault prediction models, *Empirical Software Engineering* **13**(5) (2008) 561-595.
[35] J.H. Hayes: Building a Requirement Fault Taxonomy: Experiences from a NASA Verification and Validation Research Project, In *Proc. Of the 14th International Synposium on Reliability Engineering. (ISSRE 2003)*, Denver, Colorado, USA, 2003, pp.49-59.
[36] M.S. Feather and S.L. Cornford, Quantitative risk-based requirements reasoning, *Requirement Engineering* **8**(4) (2003) 248-265.
[37] V. Wyatt, J.S. Di Stefano, M. Chapman and Edward Aycoth: A Metrics Based Approach for Identifying Requirements Risks. In *Proc. of the 28th Annual NASA Goddard Software Enginnering Workshop (SEW 2003)* Greenbelt, Maryland, USA, 2003, pp. 23-28.
[38] G. Boetticher, T. Menzies and T. Ostrand, PROMISE Repository of empirical software engineering data http://promisedata.org/ repository, (Ed.: West Virginia University, Department of Computer Science, 2007).
[39] B. Cestnick, I. Kononenko and I. Bratko. ASSISTANT 86: A Knowledge- Elicitation Tool for Sophisticated Users, In *Proceedings of 2nd European Working Session on Learning (EWSL 87)*, Bled, Yugoslavia, 1987. pp: 31-45.
[40] R.O. Duda and P.E. Hart, Pattern Classification and Scene Analysis (Wiley, *1973*).

[41] N. Friedman, D. Geiger and M. Goldszmidt: Bayesian Network Classifiers, *Machine Learning* **29**(2-3) (1997) 131-163.
[42] C.K. Chow, and C. N. Liu, Approximating discrete probability distributions with dependence trees, *IEEE Transactions on Information Theory* **14**(3) (1968) 462-467.
[43] M. Sahami, Learning Limited Dependence Bayesian Classifiers, in *Proc 2º International Conference on Knowledge Discovery and Data Mining (KDD-96)*, Portland, Oregon, USA, pp. 335-338.
[44] NASA, Software Engineering Program Software Measurement Guidebook, Report Nº: NASA-GB-001-94 (Ed.: National Aeronautics and Space Administration, 1994).
[45] Y. Ma and B. Cukic, Adequate and Precise Evaluation of Quality Models in Software Engineering Studies, In *Proceedings of the Third International Workshop on Predictor Models in Software Engineering (PROMISE,2007)*, Minneapolis, MN, USA, 2007.
[46] T. Fawcett, An introduction to ROC analysis. *Pattern Recognition. Letters.* **27**(8) (2006) 861-874.