Intelligent Information Management, **2012, 4, 291-295**
http://dx.doi.org/10.4236/iim.2012.425041 Published Online October 2012 (http://www.SciRP.org/journal/iim)

Automatic Risk Identification in Software Projects: An Approach Based on Inductive Learning

Julia Botan Machado, Silvio do Lago Pereira
Department of Information Technology, São Paulo State Technological College—FATEC/SP, São Paulo, Brazil
Email: julia.botan@gmail.com, slago@pq.cnpq.br

Received August 30, 2012; revised October 4, 2012; accepted October 16, 2012

ABSTRACT

Effective risk management is very important to increase the probability of success in software projects. Indeed, like other types of projects, software projects are also susceptible to various problems that can lead to the cancelation of their development or to the development of systems that do not meet the client's requirements. One of the main activities of risk management is the risk identification, because the list of risks generated in this activity is used all along the risk control process. Thus, this work proposes the creation of an expert system which is capable of identifying risks in software projects by using the lessons inductively learned from similar software projects already developed. By using this proposed expert system, project managers and software developers must be able to avoid errors of the past.

Keywords: Risk Management; Risk Identification; Software Engineering; Expert System

1. Introduction

There are many different definitions of risk in literature. In this work, risks are defined as future events with some probability of occurrence and a potential for loss. Every project is subject to risks and the role of a project risk manager is to anticipate the risks that can compromise the successful completion of a project and to plan how to proceed if they occur, in order to minimize the loss [1].

Effective risk management is crucial for the success of a project. Notwithstanding, risk identification is a very hard prediction problem and most software project managers still have great difficulty in performing this task. In order to overcome this difficulty, this work proposes the implementation of an expert system capable of identifying risks in software projects, by using lessons inductively learned from similar projects developed in the past. The assumption is that the experience acquired in previous projects is the main tool which developers have to aid them in the management of new similar projects. The proposed automatic risk identification procedure is based on the checklist technique [2,3], in which a checklist is used to verify whether a specific risk can or cannot occur in a project. In the system, the checklist is represented by a decision tree [4], built by an inductive learning algorithm [5,6] that works over a database containing characteristics of previous projects and their corresponding risks pointed out by experts in risk management.

The remaining sections of this paper are organized as follow: Section 2 defines the problem addressed in this work; Section 3 presents the concepts and techniques of artificial intelligence used to implement an expert system to solve that problem; Section 4 briefly discusses empirical results obtained with the system; and, finally, Section 5 presents the final conclusion.

2. Risk Management in Software Projects

Although risk management is not a linear process, often it is divided into four phases: risk identification, risk assessment, risk response planning, and risk monitoring and controlling [1]. Clearly, *risk identification* is the main phase in this process, since all the other phases depend on the correct identification of the risks. In fact, to properly manage risks, the first thing that risk managers should be able to do is to determine what risks can damage its projects and to recognize their characteristics.

Risk identification is particularly important for software projects, because this kind of project involves inherent uncertainties that are very hard to control, e.g., technological innovations and changes in the client's requirements. Indeed, due to these uncertainties, most software projects do not comply with the deadline or budget initially planned for them and, even worse, most of the products do not meet the client's expectations in terms of functionality and quality [7].

Paradoxically, in spite of the fact that most of the project failures are closely related to failures in the risk identification phase, most of project managers and software developers still perceive this activity as a useless

and hard extra work and, as soon as they can, they readily forget it [8]. This happens mainly because there are few tools that can be used to turn this activity easier [9].

To identify risks in new projects, the best practices in risk management established by PMBOK (*Project Management Body of Knowledge*) [10] strongly recommend the use of historical data, collected during the risk identification phase for similar projects developed in the past. However, although most of the organizations have a large volume of documents about previous projects, the manual extraction of useful information from this data is not an easy task.

Thus, the main contribution of this work is to propose a tool that can aid project managers and software developers in the task of risk identification and, consequently, to avoid that so important activity can become overlooked. More specifically, the proposed tool is an expert system that can identify risks in new projects, based on the history of risks already identified in similar projects.

3. The Expert System for Risk Identification

Experts in risk management advice that an effective risk identification should be performed by taking into account results of studies done by experts in risk management, as well as documents about lessons learned during the risk management process for other similar projects already concluded [11]. To do so, project risk managers should collect documents describing projects characteristics and the corresponding risks identified for them.

By using such a collection of documents, it is possible to implement a computer system that automatically identifies risks in new projects, based on the experience accumulated by human experts in the past. In fact, this is the very approach adopted in this work, as depicted in **Figure 1**. Moreover, at the end of each project, the system can update its knowledge base with the new lessons learned, such that they can be used in future projects (increasing the effectiveness and efficiency of the system).

The background on artificial intelligence and the techniques used to implement this system are succinctly introduced in the next two subsections.

3.1. Inductive Learning of Decision Trees

A *decision tree* [4] is a data structure, representing a set of classification rules, which can be used to model inductive learning and decision making abilities. The decision tree construction emulates a learning process, while its use emulates a decision making process.

A decision tree is a decision support tool, in the form of a tree graph, which models decisions and their possible consequences. A decision tree learning algorithm is a method used in data mining [12] to create a model that predicts the value of an output variable, or target variable, based on the values of input variables. A trivial example of a decision tree, with only one input variable, is depicted in **Figure 2**. In such tree, each nonterminal node corresponds to an input variable; the edges leaving a nonterminal node represent all the possible values of that input variable; and each leaf represents a value of the target variable, given the values of the input variables represented by the path from the root to the leaf.

To build a decision tree, an inductive learning algorithm needs to receive as input a set of examples of the concept that it should learn. Thus, this kind of learning is called *supervised learning*. Besides, the set of examples is often called *training dataset*. Each example is a tuple formed by the values of the input variables (always available) and also the value of the output variable, or target variable (available only in examples). The idea is that, by analyzing the correlations among the values of input and output variables, the learning algorithm can build a *hypothesis* that, afterwards, can be used to correctly predict the target variable value, in cases where only the input variable values are known. To validate the efficiency of the decision tree built, another set of brand new examples, called *test dataset*, is used. In this case, the values of the target variable in the examples are compared with those predicted by the hypothesis. The efficiency of the decision tree can be given as the ratio of the number of hits and the number of examples in the test dataset.

A tree can be built by recursively splitting a training dataset into subsets based on the values of a selected

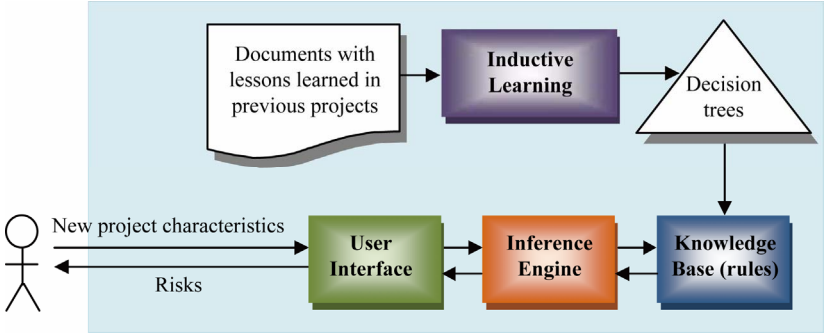

Figure 1. The architecture of the expert system.

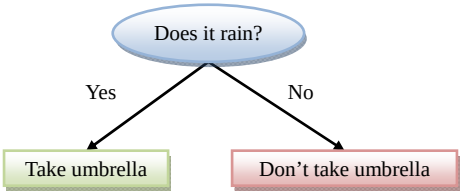

Figure 2. A decision tree for the "umbrella problem".

input variable. The recursion terminates when the subset at a node has all the same value for the target variable, or when splitting no longer enhance the predictions. This process of top-down partitioning is a kind of greedy algorithm, and is the most common strategy for learning decision trees from data. After construction and validation, the resulting decision tree can be used to emulate an efficient making decision process.

To guarantee the efficiency of the decision tree, the inductive learning algorithm uses the concepts of entropy and information gain [4] to choose the input variable to label each nonterminal node. The *entropy* is a measure based on the occurrence probability of each possible event (*i.e.*, values of the input variables). The information gain represents the estimated reduction on the entropy value resulting from the partition of the set of examples, according to the values of the input variable selected to label a node.

Formally, entropy and information gain can be defined as follow. Let E be a training dataset with examples of the form $(x_1, x_2, \cdots, x_m, y)$, where each x_i is the value of an input variable v_i, for $1 \le i \le m$, and y is the target variable value. Also, for a given input variable v_i with k possible values, let p_i be the proportion of tuples in E where the input variable v_i has value x_i. The information gain g for an input variable v_i is defined in terms of entropy h as follows:

$$h(E) = -\sum_{i=1}^{j} p_i \log_2 p_i$$

$$g(E, v_i) = h(E) - \sum_{j=1}^{k} \frac{\left|E_{v_i = x_j}\right|}{|E|} \cdot h\left(E_{v_i = x_j}\right)$$

The information gain is equal to the total entropy for an input variable if and only if, for each value of that variable, the target variable has the same value.

3.2. Expert Systems

In artificial intelligence, an *expert system* [13,14] is a computer program that emulates the ability of decision making of a human expert. In fact, by reasoning over facts and rules available in a knowledge base, an expert system is capable of solving very complex problems.

The standard architecture of an expert system (**Figure 1**) consists of a *user interface* that allows the communication with the user, a *knowledge base* that stores the knowledge about the specific application domain, and an *inference engine* that uses the available knowledge to solve problems proposed by the user.

In the expert system proposed in this work, the knowledge base is implemented as a set of decision trees (one tree for each risk) and the inference engine is a procedure that selects a proper decision tree in the knowledge base and, by reasoning with the rules encoded on this tree, decides whether a specific risk can or cannot occur, according to the projects characteristics informed by the user.

The decision trees used to populate the knowledge base of the expert system are automatically generated by an algorithm of supervised inductive learning. The inductive reasoning implemented by this algorithm allows the generation of rules about conditions that necessarily implies specific risks, by analyzing a set of documents with lessons learned in previously developed projects. These rules form, in fact, a predictive model that can be used to identify risks in new projects.

To identify risks in a new project, all that a risk manager needs to do is to access the user interface of the expert system and inform the projects characteristics. Then, the expert system should answer with a list of risks automatically identified for that project.

4. The Experiment with the Expert System

To verify the effectiveness of the proposed solution for automatic risk identification, the expert system of **Figure 1** was implemented in the Java programming language, based on the inductive tree learning algorithm ID3 [4].

This section reports some details of the experiment performed with the system and discusses some empirical results, as well.

4.1. Knowledge Base Populating

In order to decide whether a new project has a specific risk, the expert system must use a list of known risks. As said before, this list can be generated from a collection of documents describing lessons learned in previous projects. Basically, there are two types of risk: *generic risks*, which threat the most part of projects, and *specific risks*, that threat the specific project under evaluation. Generic risks can be easily detected by the expert system. On the other hand, the detection of specific risks is more complicated because, if they were not detected in previous projects, the knowledge of the expert system might be insufficient to detect their presence in a new project.

Moreover, to compare new projects with previous projects, and decide whether they are similar or not, the expert system needs to use a predefined set of characteristics which are common for all projects. These charac-

teristics must be related with risk categories, so that the expert system can reason properly.

Documents about 20 real software projects were used in the experiment performed with the implemented expert system. These documents were kindly delivered by their respective project risk managers, who also answered a questionnaire about their project characteristics and associated detected risks.

The final characterization of the projects was based on the following attributes (*input variables*):

- User involvement;
- Team experience;
- Appropriated team size (relative number);
- Staff geographical distribution;
- Team size (absolute number);
- Project priority;
- Amount of involved systems;
- Amount of involved technological platforms;
- Amount of involved databases;
- Project size (small, medium, large, huge);
- Existence of test/approval environment.

A list of seven generic risks (*output variables*), present in most of software projects, was also created:

- Risk of failing to meet the planned deadline;
- Risk of failing to meet the planned cost;
- Risk of generating a low quality product;
- Risk of ill-defined scope;
- Risk of the project cancellation;
- Risk of project postponement;
- Risk of generating a product that does not meet the needs of the user.

Based on the managers' answers for 15 projects, and each one of these generic risks, a set of documents was formatted to be taken as input (*i.e.*, training dataset) by the inductive learning module of the expert system. The resulting trees, built by this module, were used to populate the expert system knowledge base. More precisely, a set of seven decision trees, one for each one of the considered generic risks, were generated. An example of such decision trees is depicted in **Figure 3**.

Thus, given the characteristics of a specific software project, the expert system can use the rules extracted from the decision tree in **Figure 3** to inform whether this project has or not the risk of being cancelled. In this tree, each internal corresponds to an input variable, whose value is informed by the user, and each leaf corresponds to a possible value of the output variable "*risk of the project cancellation*", whose value should be predicted by the system.

4.2. Empirical Results

To validate the expert system, the remaining 5 projects were used as a test dataset.

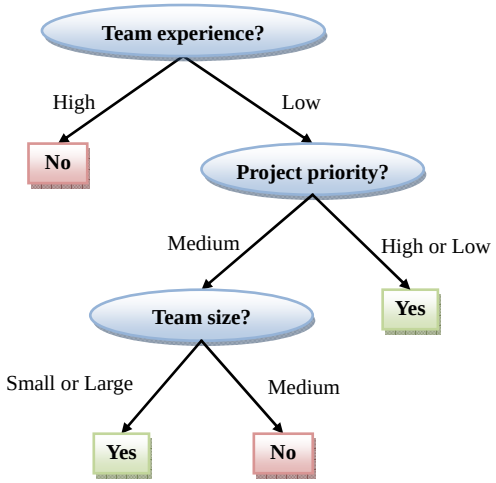

Figure 3. A decision tree for "risk of project cancellation".

The list of identified risks generated by the expert system, for each one of these projects, was compared with the results obtained through the questionnaire answered by the risk managers, referred in the last subsection.

In the total, there were performed 35 evaluations (*i.e.*, 7 risks for each one of 5 projects). It was observed that the result given by the expert system differed from that given by the corresponding project manager in only 8 evaluations. Thus, the implemented system presented a hit-rate of 77.14%. This seems to be a very promising result.

Furthermore, it is believed that, with a knowledge base populated with more historical data collected from previously developed projects, it is possible to increase this hit-rate even more.

5. Conclusions

This paper proposed an expert system capable of identifying risks in software projects, by using lessons inductively learned from similar projects developed in the past.

To evaluate the effectiveness of the implemented system, an experiment involving data about real software projects was performed. This experiment showed that the experience acquired in previous projects can really be used to automatically identify risks in new projects and to avoid repeat mistakes of the past, as well. Thus, by using a knowledge base continuously updated with lessons learned in concluded projects, the performance of the expert system will be each time better.

A frailty of the proposed system is that it only can identify risks already detected in previous projects. So, if a new project is subject to an unprecedented risk, the system fails to inform the project manager about this risk. From this observation, it is important to highlight that the proposed expert system is a tool that must be used only to help the project risk manager to perform the risk iden-

tification task. A human expert is still indispensable.

6. Acknowledgements

The authors would to thank CNPq, FAT, and especially to the project managers who contributed to this research.

REFERENCES

[1] R. K. Wysocki, "Effective Project Management: Traditional, Agile, Extreme," 5th Edition, John Wiley & Sons Ltd., Chichester, 2010.

[2] C. A. R. Morano, C. G. Martins and M. L. R. Ferreira, "Application of Techniques for the Identification of Risk in the E & P Ventures," *Engevista*, Vol. 8, No. 2, 2006, pp. 120-133.

[3] H. P. Berger, "Risk Management: Procedures, Methods and Experiences," *Reliability*: *Theory & Applications*, Vol. 2, No. 17, 2010, pp. 79-95.

[4] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, Vol. 1, No. 1, 1985, pp. 81-106. doi:10.1007/BF00116251

[5] A. Franco-Arcegaa, J. A. Carrasco-Ochoaa, G. Sánchez-Díazb and J. F. Martínez-Trinidada, "Decision Tree Induction Using a Fast Splitting Attribute Selection for Large Datasets," *Expert Systems with Applications*, Vol. 38, No. 11, 2011, pp 14290-14300.

[6] J. Gao and Z. D. Han, "New Decision Tree Algorithm with Restrained Factor Involved," *Physics Procedia*, Vol. 25, 2012, pp. 1871-1878. doi:10.1016/j.phpro.2012.03.324

[7] I. Sommerville, "Software Engineering," 9th Edition, Addison-Wesley, Boston, 2010.

[8] E. E. Odzaly and P. S. Des Greer, "Software Risk Management Barriers: An Empirical Study," *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement*, Washington, 15-16 October 2009, pp. 418-421. doi:10.1109/ESEM.2009.5316014

[9] J. Dhlamini, I. Nhamu and A. Kaihepa, "Intelligent Risk Management Tools for Software Development," *Proceedings of the Annual Conference of the Southern African Computer Lecturers' Association*, Eastern Cape, 2-11 July 2009, pp. 33-40.

[10] PMI Standards Committee, "A Guide to the Project Management Body of Knowledge," 4th Edition, Project Management Institute, 2008.

[11] Y. H. Kwak and J. Stoddard, "Project Risk Management: Lessons Learned from Software Development," Elsevier Science, Amsterdam, 2003. doi:10.1016/S0166-4972(03)00033-6

[12] S. H. Liao, P. H. Chu and P. Y. Hsiao, "Data Mining Techniques and Applications—A Decade Review from 2000 to 2011," *Expert Systems with Applications*, Vol. 39, No. 12, 2012, pp. 11303-11311. doi:10.1016/j.eswa.2012.02.063

[13] S. H. Liao, "Methodologies and Applications—A Decade Review from 1995 to 2004," *Expert Systems with Applications*, Vol. 28, No. 1, 2005, pp. 93-103. doi:10.1016/j.eswa.2004.08.003

[14] S. Lucci and D. Kopec, "Artificial Intelligence in the 21st Century: A Living Introduction," Mercury Learning and Information, Duxbury, 2012.

