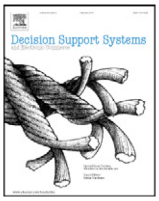
Decision Support Systems 56 (2013) 439–449

Contents lists available at ScienceDirect

Decision Support Systems

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



Software project risk analysis using Bayesian networks with causality constraints

Yong Hu [a,*], Xiangzhou Zhang [b], E.W.T. Ngai [c], Ruichu Cai [d], Mei Liu [e]

[a] Institute of Business Intelligence and Knowledge Discovery, Guangdong University of Foreign Studies, Sun Yat-sen University, Guangzhou 510006, PR China
[b] School of Business, Sun Yat-sen University, Guangzhou 510006, PR China
[c] Department of Management and Marketing, The Hong Kong Polytechnic University, Kowloon, Hong Kong, PR China
[d] Department of Computer Science, Guangdong University of Technology, Guangzhou, PR China
[e] Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA

ARTICLE INFO

Available online 8 November 2012

Keywords:
Software project risk analysis
Bayesian networks
Causality analysis
Knowledge discovery
Expert knowledge constraint

ABSTRACT

Many risks are involved in software development and risk management has become one of the key activities in software development. Bayesian networks (BNs) have been explored as a tool for various risk management practices, including the risk management of software development projects. However, much of the present research on software risk analysis focuses on finding the correlation between risk factors and project outcome. Software project failures are often a result of insufficient and ineffective risk management. To obtain proper and effective risk control, risk planning should be performed based on risk causality which can provide more risk information for decision making. In this study, we propose a model using BNs with causality constraints (BNCC) for risk analysis of software development projects. Through unrestricted automatic causality learning from 302 collected software project data, we demonstrated that the proposed model can not only discover causalities in accordance with the expert knowledge but also perform better in prediction than other algorithms, such as logistic regression, C4.5, Naïve Bayes, and general BNs. This research presents the first causal discovery framework for risk causality analysis of software projects and develops a model using BNCC for application in software project risk management.

1. Introduction

The software industry has become one of the fastest-growing industries. The global software market is estimated to have a value of US$330 billion in 2014, an increase of 36.1% since 2009 (US$ 242.4 billion) [43]. However, software development is yet a high-risk activity. The "CHAOS Summary 2009" from the Standish Group reported that the success rate of global (mainly U.S. and European) software projects is only 32% [55]. Much previous research has shown that the most important problem in software engineering is risk management, whereas technical issues are only secondary. For example, the Standish Group's report "EXTREME CHAOS" [54] summarized the recipe for software project success, that is, the CHAO 10, most of which are non-technical factors. Risk management is critical to project management; it is one of the 9 knowledge areas in project management as defined in the Project Management Body of Knowledge (PMBOK) [42] and is one of the 25 key process areas as defined in the Capability Maturity Model Integration (CMMI) [9]. McConnell believes that to obtain a 50–70% chance of avoiding time overrun, risk management only requires 5% of the total project budget [31]. These reasons highlight the urgency and feasibility of software project risk management.

In the current practice, subjective analysis or expert judgment is one of the methods often used in project risk management [15]. It is based on the experience of an expert and is thus inevitably human-intensive and obscure [16]; likewise, it generally lacks repeatability as experience is not readily shared among different teams within an organization [35]. Therefore, it is crucial to develop intelligent modeling techniques that can provide more objective, repeatable, and visible decision-making support for risk management. Among various existing intelligent modeling techniques, the Bayesian network (BN) has attracted much attention, such as those presented in refs. [1, 16, 28], due to its excellent ability in representing and reasoning with uncertainties.

Most research on software project risk analysis focuses on the discovery of correlations between risk factors and project outcomes [13, 24, 60]. At present, studies on BN-based risk analysis of software projects involve two ways of network construction: (1) experts manually specify the network to reflect expert knowledge [14, 16], and (2) automatically learn the network from observational data [27]. Since the manual method is not based on observational data, it will certainly contain expert subjective bias. The existing automatic methods for BN network learning cannot distinguish correlation from causality. For instance, the edge orientation does not necessarily indicate which risk should be controlled to change another risk. However

* Corresponding author.
E-mail addresses: henryhu200211@163.com (Y. Hu), zhxzhou@mail2.sysu.edu.cn (X. Zhang), mswtngai@inet.polyu.edu.hk (E.W.T. Ngai), cairuichu@gmail.com (R. Cai), mei.liu@njit.edu (M. Liu).

0167-9236/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.dss.2012.11.001

this limitation in existing algorithms is usually neglected. Such research models are not suitable for direct risk control.

Software project practitioners have long complained about the difficulty in determining the real and direct risks to guide the allocation of time and resources. Thus causality, rather than correlation, is of greater interest to industry experts in software project risk planning because it can determine the causal factors that directly affect project outcomes. For example, the risk of "project involving the use of new technology" may be correlated with "immature technology" because new technology is probably underdeveloped due to its unidentified bugs. Nevertheless, a new technology does not necessarily mean an immature technology. Whether we can mitigate the former risk by only focusing on the latter is not certain, and vice versa. Actually, we are advised to reduce the risks of using a new technology by referring to pilot investigations, preparing alternative technology, training of team members. National Aeronautics and Space Administration (NASA) considers that risk planning should first "make sure that the consequences and the sources of the risk are known" and "plan important risks first" [45]. The Software Engineering Institute of Carnegie Mellon University (CMU/SEI) requires the risk analysis process to satisfy the goal of "determining the source of risk", i.e., "the root causes of the risk" [18]. Hence, in risk planning, analyses of the consequences and risk sources are very important.

In this paper, we propose a novel framework for software project risk management using BNs with causality constraints (BNCC). Our primary objective is to perform a causality analysis between risk factors and project outcomes to achieve more effective risk control. Specifically, the analysis involves (1) introducing a new modeling framework for risk causality analysis to discover new causal relationships and validate existing ones (i.e., practical and/or academic expert knowledge) between risk factors and project outcomes based on historical data; and (2) constructing an empirical BN software project risk analysis model based on the framework, which can be readily used in risk planning.

Compared with other modeling algorithms such as C4.5 and Naïve Bayes, the proposed BNCC-based model has the following advantages: (1) strong interpretability — the constructed BN combines data with expert knowledge, depicts causal relationships between variables, and helps obtain better project outcomes or higher probability of project success; and (2) acceptable predictive accuracy — the final model in this study has better predictive power compared with other modeling algorithms, making the model suitable for capturing the statistical relationships between risk factors and project outcomes.

This study makes two important contributions. First, it proposes the first causal discovery framework for risk management of software projects, which builds an empirical model from real data and incorporates the causal discovery technique and expert knowledge. This risk modeling framework can be widely applied to other related domains. Second, it provides a BNCC model for risk analysis based on data from real industry software projects. The network has strong interpretability and can provide explicit knowledge (causal relationships between risk factors and project outcomes) of software projects. Subsequently, such knowledge can help in conducting effective risk analysis and further risk planning, which will result in a better implementation of software project risk management.

This paper is organized as follows. Section 2 provides a review of related literature. Section 3 describes the proposed risk model and the modeling concept. Section 4 presents the experimental results. Finally, Section 5 concludes and discusses limitations of the study.

2. Review of literature

2.1. Risk management of software projects

Risk management was first introduced to software project management by Boehm [3] and Charette [6]. According to the "IEEE Standard for Software Project Management Plans" [22], a software project is defined as a series of technical and managerial work activities that should meet the terms and conditions listed in the project agreement. Successful software project usually means that the project can be completed within the budget and given time, and meet the customers' demand for high-quality and high-performance. Wallace et al. [60] defined software project risk as a series of factors or circumstances that will be a threat to the successful completion of a software project.

Boehm [3] summarized the risk management process into two steps: risk assessment and risk control. Risk assessment involves three subsidiary steps: risk identification, risk analysis, and risk prioritization. Risk control also consists of three subsidiary steps: risk-management planning, risk resolution, and risk monitoring. Risk analysis mainly focuses on the relationships between risk factors and project outcomes, and is to prepare for further risk control. In NASA, risk analysis is the process of determining the extent of the risks, their relationships with each other, and the most important risks [45].

2.2. Risk analysis of software projects

Numerous statistical and data mining methods have been used to analyze the relationships between variables. For intelligent risk analysis of software projects, many works have employed these methods, including regression analysis [23], association rules [33], decision trees [63], fuzzy logic [62], clustering analysis [61], and neural networks [35]. Jiang and Klein [23], for instance, used multiple regression analysis to explore the various risks that significantly affect the multidimensional success of information system development. Moreno-García et al. [33] used association rules to estimate the influence of certain management policies on the software project output attributes, which include product quality, time spent, and effort exerted on the project. However, their methods have only been applied to data generated by a Software Project Simulator rather than real project data. Xu et al. [63] introduced a hybrid learning method that combines genetic algorithm and decision trees to derive optimal subsets of software metrics for risk prediction. Moreover, Xu et al. [62] developed a fuzzy expert system and illustrated how to infer the rules about software development in the early phase of its life cycle. Wallace et al. [61] performed *k*-means clustering analysis to explore the trends in risk dimensions across three clusters (i.e., low-, medium-, and high-risk projects), and then examined the influence of project characteristics (e.g., project scope, sourcing practices, and strategic orientation) on project risk dimensions. Neumann [35] combined principal component analysis with neural networks to perform software risk classification and to discriminate high-risk projects with imbalanced data sets.

Each method has its unique advantages. Regression analysis can establish the dependence between variables and can be used for prediction. Association rules can find rules that can satisfy user-specified minimum support and confidence based on (conditional) frequency counting. Decision trees are simple and easy to understand, while neural networks can capture the non-linear interdependence among variables. Fuzzy logic can aggregate the scores of risk factors into an overall project risk score based on fuzzy set theory, which is suitable for inexact risk assessment. Clustering analysis groups a set of observations into subsets based on the mutual similarity/dissimilarity of observations, without manually pre-defining specific categories. Unfortunately, none of these methods were developed to capture the causality relationships in the form of "A influences B." These methods may (unintentionally) discover some genuine cause–effect relationships, but they are unable to distinguish causality from correlation.

2.3. BN-based project risk management

BNs have a wide range of real world applications such as in diagnosis, forecasting, automated vision, sensor fusion, manufacturing control, transportation, ecosystem and environmental management [20, 56, 57].

The application of BNs on project risk management has the following advantages [16, 40, 51]: BNs can (1) model uncertainties and provide probabilistic estimates; (2) combine historical data with expert experience or prior knowledge; (3) visually model cause–effect relationships, and thus help identify risk sources, which provide explicit knowledge for risk analysis and planning; (4) be used for "what-if" analysis to explore the effect of changes in some nodes on the changes in other nodes; and (5) be used for sensitivity analysis, diagnosis, prediction, classification, and causal reasoning, among others.

Many BN-based studies have been conducted to facilitate the risk management of software projects [14, 16, 26]. For example, Fan and Yu [16] proposed a BN-based procedure using a feedback loop to predict potential risks, identify sources of risks, and advise dynamic resource adjustment. Drew Procaccino et al. [14] asked experienced practitioners to provide insights on important early non-technical issues in software development. They presented the relationships through a BN, which can be used to identify critical chains of events for success and predict the probability of success in software development.

A BN usually consists of a directed acyclic graph (DAG), which represents the network structure, and an associated set of conditional probability tables (CPTs), which are the network parameters. Three common ways to construct a BN are: (1) manually specify DAG and CPTs by expert opinion (also called expert knowledge or domain knowledge), such as in refs. [16, 34]; (2) automatically learn DAG and CPTs using various algorithms based on observational data, such as in ref. [28]; and (3) manually construct DAG by expert opinion or automatically learn DAG using expert opinion as structural constraints/restrictions, and then learn CPTs from observational data, such as in ref. [14].

Networks constructed manually by experts, such as those presented in refs. [14, 16], conform to the verified causalities (expert knowledge). This type of network is suitable for risk analysis but can inevitably result in subjective bias among different experts. Therefore, more attention is given to the ways of BN automatic learning based on observational data [28]. Even with the integration of expert knowledge as structural constraints [27], some difficulties still exist in distinguishing between correlation and causality in the final network. In software project risk management, correlation and causality are often used mistakenly for each other; causality analysis has never been given sufficient attention. As claimed by Pearl [40], "the interpretation of direct acyclic graphs as carriers of independence assumptions does not necessarily imply causation," and in statistical and artificial intelligence applications, DAGs are often used as causal interpretation tools to account for the observed data.

If risk management proceeds without distinguishing correlation from causality, then it is difficult to allocate time and resources to the key risk source (cause) of the project outcome; hence, risk control will be ineffective.

2.4. Causal discovery

Causality is the relationship between a first event (the cause) and a second event (the effect), where the second event is a consequence of the first one [12]. Usually, in a DAG, G with a set of vertices V can be used to represent causal relationships between variables, where an edge from A to B in G means that A is a direct cause of B relative to V.

Causality is important in planning and decision making in almost all fields. For example, in medicine, determining the cause of a disease helps in prevention and treatment [30]. Moreover, in software project risk management, identifying the cause for the failure of a project helps in risk control. By investigating causality, the state of the target variable can be predicted even when the states of the other factors are changed.

There are two main approaches for causal inference: intervention experiments and observational-data-based inference. In both approaches, the effects of the different values of the independent variable(s) on the dependent variable(s) are observed. In the intervention experiment approach, measurements of the system are first taken, then certain factors are manipulated, and finally additional measurements are taken using the same procedure. Conversely, the observational approach infers causality only from observational data, without any manipulation of the factors. Due to ethical, practical, and cost considerations, among others, intervention experiment may not be feasible or are too expensive in some contexts, such as in the risk management of software projects. These limitations provide the impetus to explore techniques for learning causality from observational data.

Pearl has made an important contribution to the development of causal inference both in fundamental theory and practical applications, such as those presented in refs. [38, 39, 41]. Moreover, causal discovery technology is applied in various domains, leading to prominent results. For example, Mani and Cooper [29] applied the local causal discovery algorithm to learn causalities from intensive care unit discharge summaries and identified the causal factors of clinical conditions and outcomes. In another paper [30], the authors proposed a Bayesian local causal discovery algorithm; they applied it to the Linked Birth/Infant Death data set and found six causal relationships, three of which seemed plausible. Spirtes and Cooper [49] tested a causal discovery algorithm on a database of pneumonia patients and found that the results agree strongly with the opinions of the physicians. Silverstein et al. [47] reported that learning complete causal models is essentially impossible; nevertheless, the isolated causal relationships that only involve pairs or small sets of items are easier to interpret. They explored causality, instead of mere associations, in context of market basket analysis. These previous studies have inspired us to perform causal discovery in the domain of software project risk management.

3. Methodology

3.1. Causality in BNs

BNs [10] based on graph and probability theories are a widely accepted tool that can visualize uncertain knowledge and perform efficient reasoning, given the variables and their joint probability distributions. A BN consists of two parts: a DAG, which indicates conditional (in)dependent relationships among the variables (Fig. 1), and a set of CPT, which represents the conditional probability distribution among the variables.

The faithfulness condition of Bayesian graphical theory assumes that a BN N that *faithful* to the given distribution P always exists, where P is a joint probability distribution on the variable set V, i.e., a one-to-one correspondence between the nodes in BN and the given variables in V always exists. The faithfulness condition ensures that we can always find a BN N, which contains complete and sound dependence relationships among the variables in V.

A BN consists of four kinds of basic local structures, as shown in Fig. 2. Two BN structures (N_1 and N_2) are *independence-equivalent* if they represent the same conditional independence assertions for V, where V is the set of variables in N_1 and N_2 [37]. For example, Fig. 2a, b and c are independence-equivalent BNs. For the convenient explanation of the difference between the four BN structures, we assume that variable X is not a neighbor of variable Y and there is no dependence path (i.e. block all dependence paths) between variables X

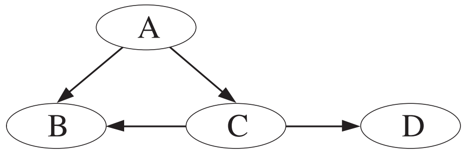


Fig. 1. A simple BN.

$$X \rightarrow Y \rightarrow Z \quad \text{(a)}$$
$$X \leftarrow Y \leftarrow Z \quad \text{(b)}$$
$$X \leftarrow Y \rightarrow Z \quad \text{(c)}$$
$$X \rightarrow Y \leftarrow Z \quad \text{(d)}$$

Fig. 2. Four kinds of basic local structures.

and *Z* through other variables. Fig. 2a, b and c imply the same assertion that variable *X* is conditionally independent of variable *Z* given variable *Y*. However, Fig. 2d is not independence-equivalent to the other three. And we refer to it as a "V-structure" or a "collider", which implies a different assertion that variable *X* is independent of variable *Z* not given variable *Y*, but variable *X* is conditionally dependent of variable *Z* given variable *Y*.

All four structures shown in Fig. 2 can represent causality, but the difference merits more discussion. Independence-equivalent BNs (i.e., Fig. 2a, b, and c) express the same dependency information, which means that they are equally faithful to the joint probability distribution of the given variables. Hence, we cannot distinguish independence-equivalent BNs based on data only without any expert judgment. In contrast, a V-structure, i.e., Fig. 2d, can be exclusively determined from data without any expert opinion because it has no independence-equivalent BN.

3.2. V-structure discovery algorithm

In this section, we present our V-structure discovery algorithm using the framework of inductive-causation [41, 58], which can find local cause–effect relationships (rather than a complete BN) between variables from observational data. We focus on local cause–effect relationships because learning complete causal models from observational data is unrealistic and essentially impossible, as claimed in ref. [47].

The proposed algorithm shown in Fig. 3 takes a data set defined on a variable set *V* as input and a list of V-structures as output. Each variable *v* in *V* is processed separately in two steps to discover the V-structures that take *v* as the center. First, we use the neighbor nodes discovery algorithm proposed in ref. [5] to determine the parents and children node set PC_v of *v*. Subsequently, we enumerate every candidate V-structure that takes *v* as the center, i.e., $v_1 \rightarrow v \leftarrow v_2$, where v_1 and v_2 belong to PC_v and they are not neighbors (i.e. v_1 does not belong to PC_{v2}, and v_2 does not belong to PC_{v1}), and then label each candidate as a genuine V-structure if v_1 and v_2 are always dependent given *v* and any subset (excluding v_1 and v_2) of PC_v.

Conditional independence tests are performed to determine the parents and children node set PC_v of variable *v* and verify the candidate V-structure $v_1 \rightarrow v \leftarrow v_2$. In this study, we use *g*-squared (also g^2) [50] instead of chi-squared (also χ^2) for conditional independence tests. Chi-squared tests have been commonly used for independence tests because the log-likelihood ratio calculations on which the *g*-squared tests are based used to be unduly laborious, while chi-squared tests are approximations of *g*-squared tests and requires less calculation effort. Given that the calculation of the log-likelihood ratio is no longer a problem, *g*-squared tests are becoming increasingly used, particularly because they are recommended in the popular statistics textbook by Sokol and Rohlf [48].

3.3. Expert knowledge constraints

Generally, structural constraints [10] of BNs include existence constraint, absence constraint, and partial ordering constraint. Given a BN *N* and two variables, *A* and *B*, of *N*, an existence constraint, "$A \rightarrow_e B$", means that there must be a direct connection from *A* to *B*; an absence constraint, "$A \rightarrow_a B$", means that there must not be a direct connection from *A* to *B*; a partial ordering constraint, "$A \rightarrow_p B$", means that if there exists a direct connection between *A* and *B*, then it must be "$A \rightarrow B$" while "$B \rightarrow A$" is not allowed.

To reveal valuable causality from observational data, imposing partial ordering constraints is necessary. On one hand, it can take advantage of the aggregation of expert knowledge and automatic network learning to identify unknown but potentially valuable causalities (main objective) and to verify the known causalities (secondary objective). On the other hand, it helps to eliminate the negative effects of noisy data (especially in the cases when samples are limited).

The key in adding partial ordering constraints is to control constraint granularity and provide authorized knowledge. Our research sets up constraints defined on risk dimensions. The reasons include but are not limited to the following. First, partial ordering constraints among risk dimensions are perspicuous and can easily obtain literature support and industry approval. However, researchers have subjective biases on partial ordering among specific risk factors. Second, partial ordering constraints between risk factors cannot discover "counter" causality, which has a contrary edge orientation compared to the expert knowledge. Setting constraints for each factor will affect the knowledge discovery ability. Third, causality is used to describe the relationship between risk factors and project outcomes. Using a

```
Input: a data set of a variable set $V$

Output: a list of V-structures in the form $v_1 \rightarrow v \leftarrow v_2$

For each variable $v$ in the given variable set $V$, the following procedure is performed to discover all cause-
effect relationships related to $v$:

(1)  Find all neighbor nodes (i.e., parent and child nodes) set $PC_v$ of $v$.

(2)  For each pair of nodes $v_1$ and $v_2$, perform the following in $PC_v$:

       i.   If $v_1$ is a neighbor of $v_2$ and vice versa, then skip ii) and iii).

      ii.   Choose the smaller of $PC_{v1} - \{v\}$ and $PC_{v2} - \{v\}$ as candidate condition set $C$.

     iii.   If $v_1$ and $v_2$ are dependent given $S$ and $v$ for all subsets $S$ of $C$, then output $v_1 \rightarrow v \leftarrow v_2$.
```

Fig. 3. V-structure discovery algorithm.

higher level of granularity (between risk dimensions) will not affect the discovery of relationships between individuals.

3.4. Modeling framework

To better implement project risk control, this research performs risk causality analysis. By combining expert knowledge, V-structure discovery algorithm, and automatic network learning, we establish a BN for risk analysis.

Expert knowledge can simplify the search for a BN structure representing a given domain with improved prediction accuracy [10]. Lauría et al. [27] demonstrated how to incorporate existing expert knowledge (or expertise) as network structural constraints in the process of automatic learning of IT implementation BN from real data. They developed a reasonable model, but their network cannot distinguish causality from correlation, which adds to the difficulty of direct and effective risk planning.

Consequently, we establish a BNCC based on real data. The general idea is as follows. First, V-structures are discovered. Next, they are connected to form a causality network (including one or more DAGs) [58]. Finally, BN automatic learning is performed based on the given structural constraints defined by the causality network and expert knowledge. The specific modeling steps include the following:

(1) collect project samples using a structured questionnaire or other equivalents;
(2) pre-process data (e.g., data cleansing and data discretization);
(3) learn V-structures from the samples and connect them to construct a causality network, which is used as existence constraints;
(4) define the partial ordering constraints between risk dimensions, according to expert knowledge; and
(5) use existence constraints defined in 3) and partial ordering constraints defined in 4) as structural constraints and automatically learn a BN based on the samples.

4. Application of the methodology

4.1. Software project risk model

Measurements of risks in software projects have been studied since the 1980s, and various risk classification frameworks, dimensions, and models have been proposed. Some of these have been used for special software projects, such as e-commerce [36] and customer relationship management system [44], among others. However, most of the existing studies [2, 4, 46, 52] lack a comprehensive and systematic framework. Boehm [4] developed a top 10 risk identification checklist. Although these risk factors are considered critical to a software project, they are not organized in a systematic framework. Barki et al. [2] proposed 35 software risks grouped into five dimensions, namely, technological newness, application size, expertise, application complexity, and organizational environment. However, they paid less attention to project management related risk factors. Schmilt et al. [46] conducted international surveys covering three different cultures to develop a software risk classification framework, which included 14 dimensions and 33 risks. They focused on analyzing the common features and differences in risk factor rankings across different cultures. Takagi et al. [52] designed a risk framework from five project viewpoints (i.e., requirements, estimates, planning, team organization, and project management activities) and identified 22 risk factors. However, their framework did not cover the risk factors related to the organization environment; these factors can significantly impact the performance of a software project [59].

Wallace et al. [59] defined 27 software risks classified into six dimensions, as shown in Table 1, and seven project performance measures classified into two dimensions, as shown in Table 2. In this research, we use this classification framework because it summarizes the previous studies, conforms to the socio-technical system theory, suitable for a wide range of software projects, and has been the topic of subsequent research [19, 21].

Table 1
Risk dimensions and risk factors (Wallace et al., 2004) [59].

Risk dimension	Abbr.	Risk factor
Organizational environment risk	*Org1*	Change in organizational management during the project
	Org2	Corporate politics with negative effects on the project
	Org3	Unstable organizational environment
	Org4	Organization undergoing restructuring during the project
User risk	*User1*	Users resistant to change
	User2	Conflict between users
	User3	Users with negative attitudes toward the project
	User4	Users not committed to the project
	User5	Lack of cooperation from users
Requirement risk	*Req1*	Continually changing system
	Req2	System requirements not adequately identified
	Req3	Unclear system requirements
	Req4	Incorrect system requirements
Project complexity risk	*Comp1*	Project involves the use of new technology
	Comp2	High level of technical complexity
	Comp3	Immature technology
	Comp4	Project involves the use of technology that has not been used in prior projects
Planning and control risk	*P&C1*	Lack of an effective project management methodology
	P&C2	Project progress not monitored closely enough
	P&C3	Inadequate estimation of required resources
	P&C4	Poor planning
	P&C5	Project milestones not clearly defined
	P&C6	Inexperienced project manager
	P&C7	Ineffective communication
Team risk	*Team1*	Inexperienced team members
	Team2	Inadequately trained development team members
	Team3	Team members lack specialized skills required by the project

4.2. Sample data set

4.2.1. Data collection and demographics

The current study was conducted in Mainland China. A total of 500 questionnaires were sent out to employees of randomly selected software companies that are listed as members of the Guangdong Software Industry Association. The names of the members of the association were arranged in random before the selection to avoid selection bias. The respondents were asked to complete the questionnaires based on their recently concluded software projects. Out of 500, 350 respondents returned the questionnaires, thus yielding a response rate of 70%. After discarding the incomplete questionnaires, 302 valid responses were obtained. The summary of the profile demographics is shown in Table 3.

The responses were not filtered because the employed risk model in this present study was meant for general software projects. Various

Table 2
Product and process performance measures (Wallace et al., 2004) [59].

Performance	Abbr.	Attribute
Product	*QS1*	The users perceive that the system meets the intended functional requirements.
	QS2	The system meets user expectations with respect to ease of use, response time, and reliability.
	QS3	The application developed is easy to maintain.
	QU1	The users are satisfied with the developed application.
	QU2	The overall quality of the developed application is high.
Process	*Time*	The system was completed within budget.
	Cost	The system was completed within schedule.

Table 3
Demographics of the respondents.

Characteristics	Frequency	Percent (%)
Level of the respondents		
CEO	14	4.64
Project manager	78	25.83
Project technical leader	52	17.22
Development team member	111	36.75
Customer manager	36	11.92
Others	11	3.64
Work experience		
Under 3 years	50	16.56
3–6 years	170	56.29
7 or above	82	27.15
Industry		
Government	44	14.57
Education	25	8.28
Finance	36	11.92
Information	86	28.48
Health	15	4.97
Manufacturing	35	11.59
Commerce	46	15.23
Insurance	3	0.99
Transportation	10	3.31
Others	2	0.66
Function points		
≤500	232	76.82
501–1,000	39	12.91
1,001–5,000	23	7.62
5,001–10,000	6	1.99
≥10,001	2	0.66

types of software projects were collected from cross-sectional data, including the development of information systems (over 70%), client–server systems, e-commerce systems, new generation of game systems, mobile applications, office automation systems, and so on. Among those samples, 25.2% of the projects are considered successful, whereas the rest are regarded unsatisfactory or canceled. The success rate corresponded with the actual software productivity, which is slightly lower than the average success rate of global software projects (32%, as reported by the Standish Group [55]). A wide variety of industries, including the government (14.57%), information industry (28.48%), manufacturing (11.59%), and commerce (15.23%), among others, are represented in the final samples (302). In terms of project scale, more than 40% of the projects have more than 10 team members and more than six months of development time. The development scale ranges from under 500 to over 10,000 function points, and the project, which has less than 500 function points, is the largest contributor (76.82%). The respondents are highly qualified individuals who can provide credible information for the study. Over 80% of the respondents are project managers (25.83%), project technical leaders (17.22%), or development team members (36.75%) with a related work experience of more than three years.

4.2.2. Data pre-processing

All factors are measured on a five-point Likert scale. We followed the methodology introduced by Lauría et al. [27] to discretize the data using a binary scheme, where s0 = “low” and s1 = “high” performance (high performance indicates low risk). According to the Standish Group [53], a software project is considered completely successful if all aspects are successful (i.e., completed on-time and on budget with all features and functions as initially specified), otherwise challenged (i.e. partially failed) or failed completely. With respect to the two output variables, we define process performance as “high” when both *Time* and *Cost* are “high”, otherwise as “low”. Similarly, we define product performance as “high” when *QS1*, *QS2*, *QS3*, *QU1*, and *QU2* are “high”, otherwise as “low”. We regard projects with “high” product performance and “high” process performance as successful.

The size of our data set does not permit greater granularity (e.g., where s0 = “high”, s1 = “medium”, and s2 = “low”). Assuming an average of three parents per variable, our binary scheme with 29 variables requires the estimation of 232 parameters (29×2^3), comparing favorably with the 302 available cases.

4.3. Expert knowledge

Wallace et al. [59] studied how software project risks affect project performance and developed their model based on project management literature and the socio-technical systems theory. Their model is well-received by colleagues and has been further studied [19, 21]. The socio-technical systems theory emphasizes on the fit between technical and social subsystems. Social subsystem risks consist of organizational environment risk and user risk, whereas technical subsystem risks consist of requirement risk and project complexity risk. Planning & control risk and team risk are grouped into project management risks.

In ref. [59], the following hypotheses were verified: (1) project management risk affects process performance and product performance; (2) process performance affects product performance; (3) social subsystem risks and technical subsystem risks affect project management risks; and (4) social subsystem risks affect technical subsystem risks. Therefore, we obtain the following partial ordering constraints among six risk dimensions:

(1) Org, User, Req, Comp, P&C, Team $\rightarrow_p$ Process, Product
(2) Process $\rightarrow_p$ Product
(3) Org, User, Req, Comp $\rightarrow_p$ P&C, Team
(4) Org, User $\rightarrow_p$ Req, Comp

4.4. Discovered causality

The V-structure discovery algorithm was applied to the collected samples. In sum, 10 V-structures were determined to cover 14 risk factors and two project outcomes. We then connected these V-structures to construct a causality network by connecting two V-structures which shares a common node. The resultant causality network has 19 edges (one duplicate causality edge was deleted), as shown in Fig. 4 (risk factors not covered by the V-structures are not shown). Two V-structures, for example, “*Req4*→*P&C3*←*Team2*” and “*Req4*→*Team1*←*Team2*” can be connected to the upper-left part of the network (Fig. 4). The most discovered causalities are in accordance with the current expert knowledge. Only three dotted edges out of 19 obtained edges showed nonconformity with the model of Wallace et al. [59], such as “*Product*→*Team3*” and “*Req2*→*User4*”, which should be reversed. The learned directions were inconsistent. Generally speaking, the product performance did not affect the skill of the team member, whereas incomplete requirements were due to the lack of user commitment. Nonconformity mainly emerged from the noisy data in the collected samples. After identifying the noisy data, three dotted edges were deleted, whereas 16 edges were retained.

Compared with the study of Lauría et al. [27], this research does not add any constraint among risk factors or risk dimensions at this stage. High conformity of most causal edges with expert knowledge shows that this causal learning method can effectively discover explicit knowledge. Noisy data lead to inconsistent dotted edges between this research and previous expert knowledge. More importantly, all edges represent causality instead of correlation, which leads to strong and direct guidance for risk control. V-structure learning cannot always generate a complete network. Hence, we need to further study a BN for risk prediction and risk analysis.

4.5. BN for software project risk analysis

To construct a model for software project risk analysis, this research establishes a complete BN based on the causality network shown in Fig. 4.

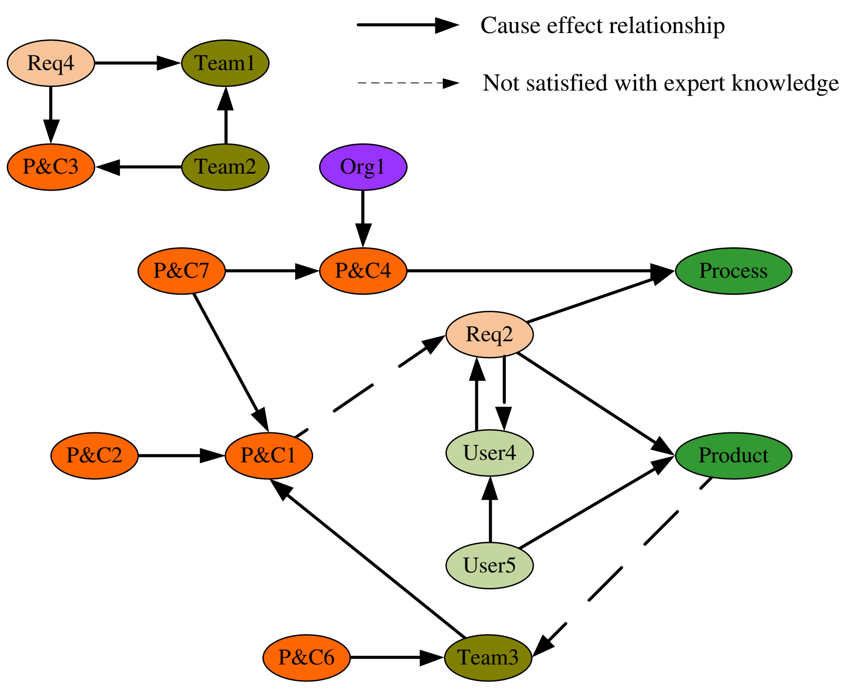


Fig. 4. Discovered causality network without constraints.

4.5.1. Final model

We apply Cheng's BN learning algorithm [7] to construct a complete model. This algorithm, an extension of the Chow and Liu's BN algorithm [8], has three phases: drafting, thickening, and thinning. Expert knowledge is added (see Section 3.4) as partial ordering constraint on the edge orientation among variables. Causal edges (see Fig. 4) are directly added as existence constraints (i.e. existing edges of the final BN). Finally, a BN based on the total samples is automatically learned, as shown in Fig. 5. Different types of line shapes represent different types of edges: the bold line represents the causal edge found by the V-structure discovery algorithm; the arrow direction of the edge represents the causal direction; and dotted line represents the correlation edge, the direction of which has no significance and can be redirected to a reverse orientation.

4.5.2. Model analysis: causality analysis

In Fig. 5, the risk factors involved in the causal edges include *User4* "Users not committed to the project," *Req2* "System requirements not adequately identified," *User5* "Lack of cooperation from users," *P&C4* "Poor project planning," and *Team3* "Team members lack specialized skills required by the project," among others. These risk factors have been studied in software project risk analysis and some have been regarded as among the top 10 risks [19, 25, 53]. To improve the "Process" and "Product" performances of the project, the abovementioned key risks must be effectively mitigated. Detailed information implied in the causality edges is described as follows.

First, we analyze the directed causality between risk factors and project performance. *Req2* is the direct common risk of "Process" and "Product" performances. *User5* is the direct reason for low "Product" performance. *P&C4* is the direct reason for low "Process" performance. These findings well coincide with academic and practical software engineering knowledge.

Second, we discuss the underlying software engineering knowledge in detail. *Req2*, *User5*, and *P&C4* (i.e., requirement risks, user risks, and planning and control risks, respectively) may directly affect project performance.

Requirement risk is widely regarded as the most important risk in a software project. The entire project is driven by requirements; thus, without proper requirement analysis to develop a complete (i.e., requirements are adequately identified, or *Req2* risk is low) and accurate set of requirements, the possibility of developing a software that "no one wants to use" increases [25]. In this study, some project managers reported that the real (i.e., most needed) requirements of a project are difficult to understand because of ineffective communication with the users. Thus, "gold-plating" [4] phenomena, i.e., setting too many unnecessary requirements, often overwhelms the real requirements, which inevitably lengthens the project schedule (i.e., increases the risk of "process" performance). It is one of the most important reasons for project failure.

User5 risk has direct and indirect effects on both "Product" and "Process" performances. "User commitment," which is affected by user cooperation through the causality "*User5*➔*User4*" in our final model, helps ensure that users are actively involved in the requirement acquisition process. It creates a sense of ownership, which in turn minimizes the risk of the users rejecting the final software [25]. Strong user commitment can even compensate for the lack of executive commitment [25]. A software project is driven by the users and not by the developers [32]. Users/customers should cooperate with project managers and play an important role in defining the software's functionality requirements. Users have the responsibility of confirming what the software should look like, what functions are required, and how the software should work [25].

Project planning can affect the project process. Demarco [11] regards "inherent schedule flaws" as one of the five core risks, which indicates that a software project is inherently difficult to schedule because of the "intangible nature and uniqueness" of software. Project managers must proactively anticipate and plan for contingency that can threaten the development process and consequently result in poor quality software delivered over-time and over-budget [25]. Project planning is mainly influenced by *Org1* "Change in organizational management during the project" and *P&C7* "Ineffective communication." As changes in the senior management or change of scopes and objectives in the business process [25], the previous well-defined project planning may need modifications, thus increasing the risk of poor planning due to a tight or fluctuated project schedule. However, due to poor internal communication, it is difficult to ensure the accurate and timely information exchange between the appropriate organizational levels and entities (e.g., levels within the development team and organization,

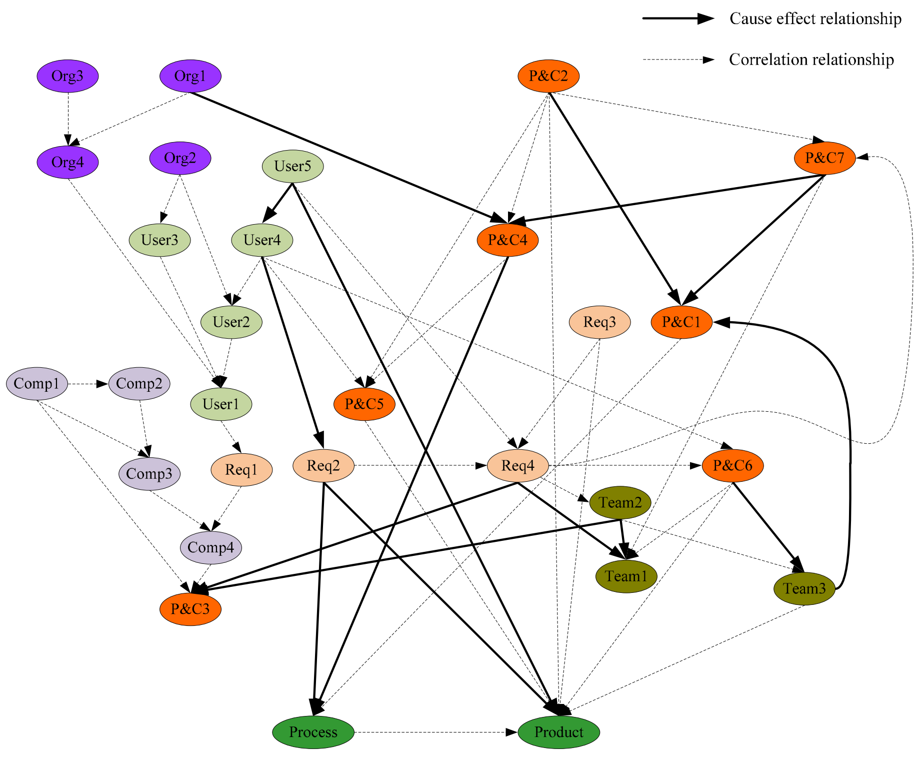


Fig. 5. The final BN with causality constraints.

within the customer organization, and especially between the developer, the customer, and the user). This will lead to vague and/or unrealistic project arrangements and scheduling, resulting in poor project planning and poor process performance.

Within the whole network, there are other causalities that highly coincide with software project risk management knowledge. For example, *Team2* “Inadequately trained development team members” is the reason for *Team1* “Inexperienced team members,” whereas *Req4* “Incorrect system requirements” is the reason for *P&C3* “Inadequate estimation of required resources.”

4.5.3. Model evaluation: prediction performance and comparison

For the model performance, this study used a 10-fold cross-validation method to evaluate the prediction accuracy of the learned BN and compared it with the classic data mining algorithms that can provide explicit knowledge (through an interpretable model or the “white-box” model), such as (logistic) regression analysis, decision trees, and the Naïve Bayes, rather than algorithms that simply provide classification, such as neural networks, and support vector machine. We used the algorithms employed in the well-known data-mining software *Weka* (available at: http://www.cs.waikato.ac.nz/ml/weka/). The results (see Table 4) show that the prediction accuracy of our proposed model is higher than that of typical data mining algorithms (e.g., C4.5, Naïve Bayes, TAN, BAN, and logistic regression analysis). The higher performance of the proposed model is attributed to the incorporation of expert knowledge and causality discovery into the Bayesian network learning process, which can significantly reduce the negative effect of trivial correlation and improve the reliability of the discovered causal relation among the variables, thus the proposed method promote prediction accuracy. Therefore, our model can also be used for risk prediction.

4.5.4. Model application: risk re-assessment

A BN can provide the reason for the change in conditional probability of one factor due to the change in the probability of another factor. This facilitates the measurement of changes in risk probability during the exploratory risk control of specific risk factors.

To verify whether causality suits the risk control better than correlation, we change the values of causal risk factors (i.e., *User5*, *Req2*, and *P&C4*) and correlation risk factors (i.e., *Req3*, *P&C1*, *P&C2*, *P&C5*, and *Team3*) separately to observe the changes in the probabilities of “Process” and “Product” performances. As shown in Table 5, the probabilities of “Process” and “Product” performances attributed to “High” improved by 29.5% and 14.7%, respectively, by controlling the causal risk factors (three in total). The probabilities of “Process” and “Product” performances attributed to “High” improved by 19.6% and 16.6%, respectively, by controlling the correlation risk factors (five

Table 4
Comparison of prediction accuracy (10-fold cross-validation).

Algorithm	Product performance	Process performance	Parameters	Remark
Our proposed model	75.15%	74.42%		
Weka_Logistic	68.87%	70.19%	default	
Weka_J48	70.86%	72.19%	default	C4.5
Weka_NaiveBayes	72.85%	71.19%	default	
Weka_BayesNet	73.18%	66.23%	SimpleEstimator -A 0.5 K2 -P 1000 -S BAYES	BAN
Weka_BayesNet	74.17%	68.54%	SimpleEstimator -A 0.5 TAN -S BAYES	TAN

Table 5
Summary of the sensitivity analysis of project performance.

	Process (%)		Product (%)	
	Low	High	Low	High
Current situation	56.7	43.3	58.8	41.2
High = 100% of User5, Req2 and P&C4	27.2	72.8	44.1	55.9
High = 100% of Req3, P&C1, P&C2, P&C5 and Team3	37.1	62.9	42.2	57.8
High = 100% of User5, Req2, Req3, P&C1, P&C2, P&C4, P&C5, Team3	25.8	74.2	27.6	72.4

in total). This implies that controlling fewer causal risk factors can result in better "Process" performance and controlling more correlation risk factors can yield comparative "Product" performance. In another case, controlling both causal and correlation risk factors (seven in total) can achieve even higher "Product" performance as well as comparative "Process" performance, while it is more costly thus entails a trade-off of the input–output ratio or return on investment (ROI). The results indicate that when we implement risk control, mastering causality is more effective than correlation: requires less risk control costs and acquires higher performances.

5. Discussion

Some issues deserve further discussion, with regard to the following aspects.

1) *Why not conduct intervention experiment?* In general, intervention experiment is more effective than the observational-data-based inference but it entails high costs. It is more appropriate and easy to perform if the goal is to discover causality at the level of software module or code because manipulating a module or modifying some lines of code is easy. In contrast, if the goal is to discover risk causalities of a software development project, using intervention experiment is difficult because some risks are uncontrollable (i.e. cannot be manipulated), or too difficult or expensive to control. The numerous risk combinations make it nearly impossible to test all potential risk causalities. Thus, intervention experiment is scarcely used in risk causality analysis of an entire software project. As mentioned above, observational-data-based inference identifies causality relationships only from data without any manipulation of the observed subjects, i.e., it relies on large-scale and high-quality samples rather than expensive or unrealistic experiments. Therefore, this study employs the observational-data-based inference approach.
2) *Incorporating expert knowledge constraint*. Learning with expert knowledge constraints is an important advantage of BNs. It can lower the complexity of network learning and prevent the BNs from deviating from common/well-known knowledge due to noisy data. However, subjective bias may influence the network due to the excessive use of expert knowledge constraints, and the results may not agree with the observational data. Hence, in this study, only verified partial ordering constraints among risk dimensions are used.
3) *Focusing on the critical software project risks*. Three critical risk factors were identified, including *Req2* "System requirements not adequately identified," *User5* "Lack of cooperation from users," and *P&C4* "Poor project planning." All of them can directly impact the "Process" and/or "Product" performances of the project. The superiority of mitigating these risks in risk control is exemplified in the risk re-assessment experiment. To increase the probability of project success, there risk factors should be continuously monitored, being paid more attention and resources than other risk factors (that involve in the correlation edges of the final BN shown in Fig. 5).
4) *Causality vs. correlation*. Compared with the correlation within general BNs, the causality in our model is more precise. The direction of causal edges is meaningful; hence, the effect/consequence can be influenced by controlling the cause. General BNs do not have this feature because their edge orientation is somewhat meaningless (e.g., correlation is bidirectional.). Thus, the correlation within general BNs is not quite suitable for risk control. Similarly, regression models just capture the correlation between variables.
5) *Comparison with other risk analysis models:* First, the causality knowledge in our model is learned from the data (objective) and not derived from the individual experience of project managers/experts (subjective). Therefore, our model is more reliable than other BNs that incorporate subjective knowledge. Second, the statistical methods for risk analysis are usually used to test hypotheses rather than discover causality; moreover, these methods cannot be used for prediction to a certain extent. For example, the structural equation modeling (SEM) focuses on the "yet to be proven" knowledge. It begins with hypothesis formulation of the underlying causality and then collects data for verification. In contrast, our approach is to collect data and then discover causality from the data, not postulating any hypothesis on causality. In addition, our model can analyze and predict project risks, demonstrating the ability to quantify the causal effect of risk factors on the project outcomes. For instance, if a certain risk appears, the probabilities of different outcomes can be reasonably predicted. Third, while regression models just capture the correlation between variables, our model emphasizes on the causality between variables.

6. Conclusions and limitations

To perform better risk analysis and risk planning, discovering causality between risk factors and project outcomes in risk management is important. This study proposes a V-structure discovery algorithm and establishes a BN with causality constraints. The proposed risk modeling framework is a completely new approach, suitable for solving similar risk management problems in other fields. And we provide an application case of software project risk analysis and control.

A large sample data was collected and an empirical BNCC model was established. Most causal edges correspond to current expert knowledge, which means that causal learning method can effectively discover explicit knowledge. The model can interpret usable explicit knowledge (risk–risk and risk–output causality) for risk planning in the risk management of software projects. At the same time, the prediction accuracy is comparable with other intelligent algorithms. The model is beneficial in merging risk analysis and risk control to help implementation of risk management.

This study could significantly contribute to academics and practitioners by establishing a BNCC model for risk analysis of software projects. This type of study has not been previously undertaken in the field of software project risk management; so it is hoped that this study will trigger a series of related investigations. In future work, a more complete and integrated decision support system with BNCC can be developed to support project managers in making decisions for risk (response) planning, e.g. ref. [17]. However, this study has specific limitations. First, the proposed algorithm cannot guarantee that a complete causal BN (i.e. each edge is a causal edge) can be constructed from the data. Due to the sample limitation, the causalities found could only construct a sparse/partial causality network. The more samples are added to the research, the more comprehensive a network could be found. Second, the proposed algorithm can only find a subset of the underlying causalities, i.e., only the kind shown in Fig. 2d not those shown in Fig. 2a, b and c. The latter three kinds of causalities require intervention experiments to verify.

Acknowledgements

This research was partly supported by the National Natural Science Foundation of China (71271061, 70801020 and 61100148), the Science and Technology Planning Project of Guangdong Province, China (2010B010600034), the Business Intelligence Key Team of Guangdong University of Foreign Studies (TD1202), and the Natural Science Foundation of Guangdong province (S2011040004804).

References

[1] C. Bai, Bayesian network based software reliability prediction with an operational profile, Journal of Systems and Software 77 (2) (2005) 103–112.
[2] H. Barki, S. Rivard, J. Talbot, Toward an assessment of software development risk, Journal of Management Information Systems 10 (2) (1993) 203–225.

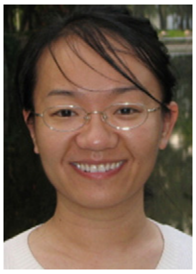
[3] B.W. Boehm, Software Risk Management, IEEE Computer Society Press, Les Alamitos, CA, 1989.
[4] B.W. Boehm, Software risk management: principles and practices, IEEE Software (1991) 32–41.
[5] R. Cai, Z. Zhang, Z. Hao, BASSUM: a Bayesian semi-supervised method for classification feature selection, Pattern Recognition 44 (4) (2011) 811–820.
[6] R. Charette, Software Engineering: Risk Analysis and Management, McGraw-Hill, Inc., New York, NY, 1989.
[7] J. Cheng, R. Greiner, J. Kelly, D. Bell, W. Liu, Learning Bayesian networks from data: an information-theory based approach, Artificial Intelligence 137 (1–2) (2002) 43–90.
[8] C. Chow, C. Liu, Approximating discrete probability distributions with dependence trees, IEEE Transactions on Information Theory 14 (3) (2002) 462–467.
[9] CMMI Product Team, Capability Maturity Model Integration (CMMI ^{SM}) Version 1.1, CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1), 2002.
[10] L. de Campos, J. Castellano, Bayesian network learning algorithms using structural restrictions, International Journal of Approximate Reasoning 45 (2) (2007) 233–254.
[11] T. DeMarco, T. Lister, Waltzing with Bears: Managing Risk on Software Projects, Dorset House Publishing, New York, NY, USA, 2003.
[12] Dictionary.com, "Causality", in: Collins English Dictionary - Complete & Unabridged, 10th Edition, (HarperCollins Publishers), 2012.
[13] J. Drew Procaccino, J. Verner, S. Overmyer, M. Darter, Case study: factors for early prediction of software development success, Information and Software Technology 44 (1) (2002) 53–62.
[14] J. Drew Procaccino, J. Verner, M. Darter, W. Amadio, Toward predicting software development success from the perspective of practitioners: an exploratory Bayesian model, Journal of Information Technology 20 (3) (2005) 187–200.
[15] S. Du, M. Keil, L. Mathiassen, Y. Shen, A. Tiwana, Attention-shaping tools, expertise, and perceived control in IT project risk assessment, Decision Support Systems 43 (1) (2007) 269–283.
[16] C. Fan, Y. Yu, BBN-based software project risk management, Journal of Systems and Software 73 (2) (2004) 193–203.
[17] C. Fang, F. Marle, A simulation-based risk network model for decision support in project risk management, Decision Support Systems 52 (3) (2012) 635–644.
[18] E. Hall, Managing Risk: Methods for Software Systems Development, Addison-Wesley Reading, MA, 1998.
[19] W. Han, S. Huang, An empirical analysis of risk components and performance on software projects, Journal of Systems and Software 80 (1) (2007) 42–50.
[20] D. Heckerman, A. Mamdani, M. Wellman, Real-world applications of Bayesian networks, Communications of the ACM 38 (3) (1995) 24–26.
[21] S. Huang, W. Han, Exploring the relationship between software project duration and risk exposure: a cluster analysis, Information Management 45 (3) (2008) 175–182.
[22] IEEE Std 1058–1998, IEEE Standard for Software Project Management Plans, 1998.
[23] J. Jiang, G. Klein, Risks to different aspects of system success, Information Management 36 (5) (1999) 263–271.
[24] J. Jiang, G. Klein, Software development risks to project effectiveness, Journal of Systems and Software 52 (1) (2000) 3–10.
[25] M. Keil, P. Cule, K. Lyytinen, R. Schmidt, A framework for identifying software project risks, Communications of the ACM 41 (11) (1998) 83.
[26] E.J.M. Lauría, P.J. Duchessi, A Bayesian belief network for IT implementation decision support, Decision Support Systems 42 (3) (2006) 1573–1588.
[27] E. Lauría, P. Duchessi, A methodology for developing Bayesian networks: an application to information technology (IT) implementation, European Journal of Operational Research 179 (1) (2007) 234–252.
[28] E. Lee, Y. Park, J.G. Shin, Large engineering project risk management using a Bayesian belief network, Expert Systems with Applications 36 (3) (2009) 5880–5887.
[29] S. Mani, G. Cooper, Causal Discovery from Medical Textual Data, in: J.M. Overhage (Ed.), The AMIA Annual Fall Symposium 2000, Hanley & Belfus, Inc., 2000, pp. 542–546.
[30] S. Mani, G.F. Cooper, Causal Discovery Using a Bayesian Local Causal Discovery Algorithm, in: M. Fieschi, E. Coiera, Y.-C.J. Li (Eds.), Medinfo, IOS Press, 2004, pp. 731–735.
[31] S. McConnell, Software Project Survival Guide: How to Be Sure Your First Important Project Isn't Your Last, Microsoft Press, Redmond, WA, 1997.
[32] F. McFarlan, Portfolio approach to information systems, Harvard Business Review 59 (5) (1981) 142–150.
[33] M. Moreno García, I. Román, F. García Peñalvo, M. Bonilla, An association rule mining method for estimating the impact of project management policies on software quality, development time and effort, Expert Systems with Applications 34 (1) (2008) 522–529.
[34] S. Nadkarni, P.P. Shenoy, A causal mapping approach to constructing Bayesian networks, Decision Support Systems 38 (2) (2004) 259–281.
[35] D. Neumann, An enhanced neural network technique for software risk analysis, IEEE Transactions on Software Engineering 28 (9) (2002) 904–912.
[36] E. Ngai, F. Wat, Fuzzy decision support system for risk analysis in e-commerce development, Decision Support Systems 40 (2) (2005) 235–255.
[37] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, 2nd ed. Morgan Kaufmann, San Francisco, California, USA, 1991.
[38] J. Pearl, Causal diagrams for empirical research, Biometrika 82 (4) (1995) 669–710.
[39] J. Pearl, From Bayesian Networks to Causal Networks, in: Mathematical Models for Handling Partial Knowledge in Artificial Intelligence, Plenum Press, New York, USA, 1995, pp. 157–181.
[40] J. Pearl, Causality: Models, Reasoning, and Inference, Cambridge University Press, Los Angeles, USA, 2000.
[41] J. Pearl, T. Verma, A theory of inferred causation, Studies in Logic and the Foundations of Mathematics 134 (1995) 789–811.
[42] Project Management Institute (PMI), A Guide to the Project Management Body of Knowledge (PMBOK Guide), 4th ed. Project Management Institute, Newtown Square PA, USA, 2008.
[43] Research and Markets, Software: Global Industry Guide, , 2010.
[44] T. Roh, C. Ahn, I. Han, The priority factor model for customer relationship management system success, Expert Systems with Applications 28 (4) (2005) 641–654.
[45] L. Rosenberg, T. Hammer, A. Gallo, Continuous Risk Management at NASA, presented at the Applied Software Measurement/Software Management Conference, San Jose, CA, USA, 1999.
[46] R. Schmidt, K. Lyytinen, M. Keil, P. Cule, Identifying software project risks: an International Delphi Study, Journal of Management Information Systems 17 (4) (2001) 5–36.
[47] C. Silverstein, S. Brin, R. Motwani, J. Ullman, Scalable techniques for mining causal structures, Data Mining and Knowledge Discovery 4 (2) (2000) 163–192.
[48] R. Sokol, F. Rohlf, Biometry: The Principles and Practice of Statistics in Biological Research, WH Freemand and Company, New York, 1995.
[49] P. Spirtes, G. Cooper, An Experiment in Causal Discovery using a Pneumonia Database, in: Artificial Intelligence and Statistics, Morgan Kaufmann Publishers, 1999, pp. 162–168.
[50] P. Spirtes, C. Glymour, R. Scheines, Causation, Prediction, and Search, 2nd ed. The MIT Press, New York, USA, 2000.
[51] I. Stamelos, L. Angelis, P. Dimou, E. Sakellaris, On the use of Bayesian belief networks for the prediction of software productivity, Information and Software Technology 45 (1) (2003) 51–60.
[52] Y. Takagi, O. Mizuno, T. Kikuno, An empirical approach to characterizing risky software projects based on logistic regression analysis, Empirical Software Engineering 10 (4) (2005) 495–515.
[53] The Standish Group, The CHAOS Report, , 1994.
[54] The Standish Group, EXTREME CHAOS, , 2001.
[55] The Standish Group, New Standish Group Report Shows More Project Failing and Less Successful Projects, The Standish Group, Boston, Massachusetts, 2010.
[56] F. Ülengin, S. Önsel, Y. İlker Topçu, E. Aktaş, Ö. Kabak, An integrated transportation decision support system for transportation policy decisions: the case of Turkey, Transportation Research Part A: Policy and Practice 41 (1) (2007) 80–97.
[57] L. Uusitalo, Advantages and challenges of Bayesian networks in environmental modelling, Ecological Modelling 203 (3–4) (2007) 312–318.
[58] M.A.J. van Gerven, P.J.F. Lucas, T.P. van der Weide, A generic qualitative characterization of independence of causal influence, International Journal of Approximate Reasoning 48 (1) (2008) 214–236.
[59] L. Wallace, M. Keil, A. Rai, How software project risk affects project performance: an investigation of the dimensions of risk and an exploratory model, Decision Sciences 35 (2) (2004) 289–321.
[60] L. Wallace, M. Keil, A. Rai, Software project risks and their effect on outcomes, Communications of the ACM 47 (4) (2004) 68–73.
[61] L. Wallace, M. Keil, A. Rai, Understanding software project risk: a cluster analysis, Information Management 42 (1) (2004) 115–125.
[62] Z. Xu, T. Khoshgoftaar, E. Allen, Application of fuzzy expert systems in assessing operational risk of software, Information and Software Technology 45 (7) (2003) 373–388.
[63] Z. Xu, B. Yang, P. Guo, Software risk prediction based on the hybrid algorithm of genetic algorithm and decision tree, Communications in Computer and Information Science 2 (5) (2007) 266–274.

Dr. Yong Hu is currently an Associate Professor and Chair in the Department of E-commerce, and Director of Institute of Business Intelligence and Knowledge Discovery at the Guangdong University of Foreign Studies and Sun Yat-Sen University. He received his B.Sc in Computer Science, M.Phil and Ph.D. in Management Information Systems from Sun Yat-Sen University. His research interests are in the areas of business intelligence, quantitative investment, software project risk management, e-commerce and decision support systems. He has published works in a number of journals and conferences such as DSS, ESWA and IEEE ICDM. Dr. Hu's research is supported by the National Natural Science Foundation, the Science and Technology Planning Project of Guangdong Province.

Xiangzhou Zhang is a Ph.D. student in Sun Yat-sen University and working as an assistant researcher in Institute of Business Intelligence and Knowledge Discovery at the Guangdong University of Foreign Studies and Sun Yat-sen University. He has received his B.S. degree in Computer Science from Sun Yat-Sen University, and M.S. degree in Management from Guangdong University of Foreign Studies. His research interests include data mining, quantitative investment, software project risk management, and business intelligence.

Prof. Eric Ngai is a Professor in the Department of Management and Marketing at The Hong Kong Polytechnic University. His current research interests are in the areas of E-commerce, Supply Chain Management, Decision Support Systems and RFID Technology and Applications. He has published papers in a number of international journals including MIS Quarterly, Journal of Operations Management, Decision Support Systems, IEEE Transactions on Systems, Man and Cybernetics, Information & Management, Production & Operations Management, and others. He is an Associate Editor of European Journal of Information Systems and serves on editorial board of three international journals. Prof.Ngai has attained an h-index of 22, and received 1490 citations, ISI Web of Science.

Dr. Ruichu Cai received his B.S. in Applied Mathematics and PhD in Computer Science from South China University of Technology in 2005 and 2010, respectively. He is currently an Assistant Professor in Department of Computer Science, Guangdong University of Technology, Guangzhou, P.R. China and State Key Laboratory for Novel Software Technology, Nanjing University, P.R. China. He was visiting student of National University of Singapore in 2007–2009. His research interests cover a variety of different topics including data mining, decision support systems, causal inference, association rule mining, and feature selection. He has published in a number of journals and conferences, such as Pattern Recognition, IEEE TKDE, and SIGMOD. Dr. Cai's Research is supported by the National Natural Science Foundation.



Dr. Mei Liu is currently an Assistant Professor in the Department of Computer Science at New Jersey Institute of Technology. She received her Ph.D. degree in computer science from the University of Kansas, Lawrence, USA and completed her postdoctoral training as an NIH-NLM research fellow in the Department of Biomedical Informatics at Vanderbilt University, Nashville, USA. Her research interest includes data mining, machine learning, text mining, decision support systems, quantitative investment, and medical informatics. She has published a number of papers in conferences and journals such as Bioinformatics, JAMIA, ESWA, EURASIP Journal on Applied Signal Processing, BMC Bioinformatics, PLoS ONE, and IEEE ICDM.