International Journal of Information Technology & Decision Making
Vol. 15, No. 6 (2016) 1391–1412
© World Scientific Publishing Company
DOI: 10.1142/S0219622016500401

A Bayesian Networks-Based Risk Identification Approach for Software Process Risk: The Context of Chinese Trustworthy Software

Jianping Li[*,†,‡,**], Minglu Li[†,§,††], Dengsheng Wu[†,‡‡],
Qianzhi Dai[†,¶,§§] and Hao Song[‖,¶¶]

[*]*College of Economics and Management*
China Jiliang University, Hangzhou 310018, P. R. China

[†]*Institute of Policy and Management*
Chinese Academy of Sciences
Beijing 100190, P. R. China

[‡]*Key Laboratory of Big Data Mining and Knowledge Management*
Chinese Academy of Sciences
Beijing 100190, P. R. China

[§]*Bureau of Planning, National Natural Science Foundation of China*
Beijing 100085, P. R. China

[¶]*School of Economics, Hefei University of Technology*
Hefei 230009, P. R. China

[‖]*School of Mathematics and Quantitative Economics*
Shandong University of Finance and Economics
Jinan 250014, P. R. China

[**]*ljp@casipm.ac.cn*
[††]*liml@nsfc.gov.cn*
[‡‡]*wds@casipm.ac.cn*
[§§]*qianzhi@casipm.ac.cn*
[¶¶]*songhaomouse@163.com*

Published 7 September 2016

For all of the many advantages and enormous benefits that information technology has brought to us, it has subjected to increasing risks that need enough attention. Bayesian network (BN) is an important probabilistic inference approach to support reasoning under uncertainty. This paper describes how BN is applied to quantify the occurrence probability of software process risk factors and the influence strength among the process risk factors in the context of Chinese trustworthy software. The information of 52 factors was obtained through a questionnaire survey of 93 project managers in five high Capability Maturity Model Integration (CMMI) level software companies. The focus of this paper is to present the key risk checklist and good timing for process risk control to improve software process risk management. Special effort has been put on the description of the experimental study, which provides the top 20 key risk factors in software process and critical software sub-processes for process risk management. The findings

[‡‡]Corresponding author.

can provide the key risk checklist to software risk manager for risk identification and decision-making in process risk management. This is a general approach and, as such, it can be applied to a certain software project or some software enterprises with updated data.

Keywords: Software process risk; risk identification; risk correlation; Bayesian network; trustworthy software.

1. Introduction

As a modern information-based society, the world is seriously dependent on software systems that control key infrastructures, such as energy, communications, finance, government and defence. Software has become a pervasive enabling element in improving productivities and fostering industries. However, the strong dependency on software carries significant risks that would arise in large part from inadequate trustworthiness in software. Over the past decades, the trustworthiness of software has become an important issue for developers as well as users. It is more important than ever to manage and ensure trustworthiness of software products. The significance of, and need for, research on trustworthiness of software has been affirmed by governments, enterprises and academia.[1–6]

Research on trustworthiness of software in the past has addressed two major issues: the definition of trustworthy software, and how to produce trustworthy software. Bernstein[7] and Wilhelm[8] defined trustworthiness from the software industry's perspective, and they have also outlined some opinions about trustworthy software research approaches. In most of the industry sectors, there are models for measuring product quality and security according to various index settings.[9–11]

To obtain the trustworthy software, as the source of software product, high-quality software process or software engineering has become a hot issue in software industry. Because the increasing demands for trustworthy software, software process management would be improved to a large extent over the next two decades.[12] Producing trustworthy software is a multifaceted problem of software engineering, security engineering, and risk management. De Bakker *et al.* demonstrated a strong anticipated relationship that existed between risk and project success in the literature.[13] In order to obtain trustworthy software, a strict process risk identification and control is essential.[14] Producing trustworthy software starts with an outstanding software engineering practices, and is augmented with sound technical practices and supported by risk management practices.[15]

In this paper, we focus on process risk identification for trustworthy software risk management. To be specific, we extend the research work to the sub-process risks in order to obtain the sub-process risk data and enhance the operability of risk management during software process. Based on the analysis of the relation characteristics of software risk factors, the risk influence network is adopted to depict the risk transfer and accumulation, and it shows strong ability to explain the process software relation for risk identification. Data obtained through a questionnaire survey of

93 project managers in five CMMI level software enterprises are used in our Bayesian network (BN) parameter learning. Then the network with parameters can quantify the occurrence probability of software process risk factors and the influence strength among the process risk factors. Based on the results, the key risk checklist can be obtained to software risk manager for decision-making in process risk management. Our approach can be generally applied to some software projects or some software enterprises with updated data.

This paper is organized as follows. In Sec. 2, literature review is provided. In Sec. 3, we discuss the definition of the trustworthy software and the relationship between software process and product. In Sec. 4, we introduce the process risk management for the trustworthy software including risk identification and risk control. Based on BN and software process model, the process risk identification approach is proposed in Sec. 5. Furthermore, we utilize this approach to analyze software process risk factors with the survey data from Chinese software industry in Sec. 6. At the end, some conclusions and feasible directions are provided.

2. Literature Review

To obtain trustworthy software, risk identification is the first step in software risk management. Software risk identification is one of the important issues in software risk management. A number of approaches on software risk identification have been proposed to identify risk factors and analyze the relationships between them. According to the approaches, relevant research can be mainly classified as follows:

(1) Subjective analysis (or expert judgment). For example, Du *et al.* collected data from 118 IT project experts and 140 novices through a role-playing experiment to analyze the attention-shaping tools and their interactions with expertize and perceptions of control on individual decision-making about risks in IT projects.[16] It is one of the most common approaches used in the current practice for project risk management.[16] But this kind of approach has a limitation, that is, it is generally lacks repeatability since the experience of experts is human-intensive and obscure.[17]

(2) Statistical analysis. For instance, Wallace *et al.* developed an exploratory model based on structural equation modeling (SEM) for investigating relationships between software project risks and project performance.[18] Jiang and Klein proposed a model using principal component analysis (PCA) to analyze the relationships between system success measures and risk factors.[19] In general, these researches using various statistical analysis tools aim to discover the correlation or causality between risk factors and project outcome, but they usually ignore the project process in risk analysis.

(3) Data mining. Many works have employed data mining-based approaches, such as association rules,[20] decision trees,[21] fuzzy theory,[22] clustering analysis,[23] neural networks[24] and BN,[25] to analyze the relationships among project risk factors and predict dynamic risk levels. Each data mining-based approach has its unique advantages.[25] Compared with other data mining-based approaches, BN has the

following advantages[17,25]: (1) it can model uncertainties and give probabilistic estimates; (2) the expert experience (or prior knowledge) can be considered; (3) it can analyze cause–effect relationships visually, which would be helpful to identify risk sources; (4) "what-if" analysis can be made so as to investigate the effect of changes in some nodes on the changes in other nodes; (5) it can also be used for making other analysis, such as diagnosis, causal reasoning, sensitivity analysis, classification, prediction. Because of the advantages of BN, it has been used as an important tool to exploit different information, including deterministic or probabilistic information of complex relations among variables. The proposal of using BN as a framework for reliability analysis has given rise to a research trend comparing with classical reliability formalisms. Applications of BN comprise a large range of risk management researches.[26–28] In particular, a lot of research interest exist in the use of BN models for software system reliability or software engineering risk management.[25,29,30] Fan and Yu proposed a Bayesian belief networks-based approach to predict potential risks, identify risk sources, and advise dynamic resource adjustment.[17] Lauría and Duchessi demonstrated how to create a BN from real-world data on Information Technology (IT) implementations, and also displayed the resulting BN and describes how it can be incorporated into a DSS to support "what-if" analyses about IT implementations.[30] Lauría and Duchessi provided a methodology for building an IT implementation BN from client-server survey data, and also demonstrated how to use the BN to predict the attainment of IT benefits, given specific implementation characteristics and activities.[31] Lee *et al.* presented a scheme for large engineering project risk management using BN and applies it to the Korean shipbuilding industry.[32] Fenton *et al.*[33] and de Melo and Sanchez[34] utilized BN to predict software defects and software maintenance project delays, respectively. Li *et al.* applied BN to evaluate project process risks.[35] Zhang *et al.* estimated the navigational risk of the Yangtze River by incorporating of formal safety assessment and BN.[36] In conclusion, these BN-based studies ignore the project process in risk identification phase as well as the risk correlation among different sub-process risk factors.

In this paper, we select BN as the technique tool to make risk identification for trustworthy software risk management. To get a more reliable result, risk analysis studied in this paper has two specific characteristics: (1) the software project process is opened, which is divided into several project sub-processes. Instead of the project risk of the whole software process, we redefine risk occurrence in different sub-processes to make the model more meaningful and practical. Previous software risk management studies emphasize risk identification and control for the whole software process, but do not redefine to each individual sub-process, it makes many risk identifying approaches difficult to implement in practice. In other words, software process risk management software is risk management for every sub-process, but the whole process risk checklist cannot be competent for helping identify the risk in different sub-process. Because of the risk transfer and accumulation effect, the same type of risk at different stages has effects on the final products in the different levels. (2) Risk correlation among various process risk factors is considered.

Because of risk correlation, the risk transfer would be possible and the risk accumulation effect could be amplified. Besides, risk correlation models for risk management should not only capture the complex process risk correlation characteristics, but also be easy to utilize, intuitive, and be computationally efficient. BN can consider the both specific characteristics simultaneously and have many other unique advantages as mentioned above, thus this paper utilizes BN to quantify the occurrence probability of software process risk factors and the influence strength among the process risk factors.

3. Trustworthy Software and Software Process

Though trustworthy software has been a hot issue recently, however, the concept of software trustworthiness is still in an ambiguous state. In the National Software Studies Workshop on Trustworthy Software at California, more than 20 researchers in the field proposed their understandings about the trustworthiness of software. Their viewpoints can be mainly divided into two categories. One is "Trustworthy software is stable software. It is sufficiently fault-tolerant that it does not crash at minor flaws and will shut down in an orderly way in the face of major trauma."[6] The other is "software for which there is a very high degree of confidence that this software will do exactly what it is required to do or what it is advertised to do. No more and no less."[37] The National Institute of Standards and Technology (NIST) defines trustworthiness as "software that can and must be trusted to work dependably in some critical function, and failure to do so may have catastrophic results, such as serious injury, loss of life or property, business failure or breach of security". For a product, trustworthiness is the level or degree of the users can trust its performance. In the case of software, trustworthiness can be considered as the degree of confidence about the software satisfying requirements. Based on the above viewpoints, Li *et al.* defined the software trustworthiness as it can be trusted to fully meet the requirements which may be expected from a particular software component, application, system, or network.[35] Many attributes, such as data security, stability, security, quality, privacy and safety, are involved in software trustworthiness. Each individual attribute represents a source of vulnerabilities that can threaten the software trustworthiness. Therefore, these attributes can be treated as the measurement dimensions of software trustworthiness. In summary, all of these definitions are reasonable and they are just from different perspectives. In this paper, we adopt the definition of Li *et al.*[35] because their definition is based on the typically viewpoints and more qualified the situation in this paper.

Software is the outcome of the software process. Software process is a structural framework within which a software product is developed. Software companies produce software through a series of actions and methods, and this series of sub-processes comprise the main parts of the software process. It is the integration of methods and standards for improving and mastering processes, supporting processes and management processes, throughout the software's work cycle.

The relationship between software and software process emphasizes that software trustworthiness is highly dependent on product planning, requirements, technical strategies, management decisions and especially risk management throughout the software process's work cycle. According to the software standard of ISBSG (The International Software Benchmarking Standards Group), the waterfall process model is used in our research, which includes: planning process; requirement process; design process; development process; testing process; and implementation process. After each sub-process, the process proceeds to the next sub-process, just as builders do not revise the foundation of a house after the framing has been erected.

For ensuring high trustworthiness, the software process should incorporate strict quality control and efficient risk management measures. In addition, to improve the overall development process performance, excessive costs and delays in software development have to be avoided. The trustworthiness depends on not only risk control in the software process, but also quality management in the software development process. High quality of process risk management could ensure the software process producing expected trustworthy products that satisfy users' requirements. Therefore, key risk factors and key software sub-process management become an important problem for software project managers.

4. Risk Identification for Trustworthy Software

Theoretically, a risk factor is composed of two components: the likelihood that a loss will occur and the significance or magnitude associated with the possible loss due to the risk.[38,39] The two risk components are assessed differently in approaches. One is quantitative approaches that estimate the occurrence probability of risk factors and the absolute magnitudes of potential losses. The other is subjectively qualitative risk assessments such as the involved practitioners' consideration, assess, and prioritizing risks without using of formal metrics. Independent of adopted approaches, risk assessments may influence their perception on project risks and their risk management decision makings.

Risk management for ensuring software trustworthiness is a structured approach for managing risk factors that can potentially threaten trustworthiness of the software product. It is a continuous process that aims to systematically treat risks throughout the software project lifecycle. The objective of risk management is to minimize the following impact of potentially negative events and manage the risks that might influence the trustworthiness of software.

Software risk management is a relatively new discipline whose objectives are to identify, address, and eliminate software risk items before they become either threats to successful software operation or major sources of expensive software rework. In risk management framework of Boehm, software risk management includes 4 parts: risk identification; risk analysis; risk prioritizations and risk management planning; risk resolution; and risk monitoring.[38] Wallace *et al.* identified six dimensions of software project risk which provided better understanding of software project risks and how they will affect project performance.[18] Software risk assessment requires to identify the risk items (or factors) that must be controlled to keep the project on track.[40] Subsequently, software practitioners apply various risk resolution tactics to address and control the identified risks.[41]

In common, the checklist approach is a quick and low cost way of identifying risk factors and assessing the risk exposure of the software process. Risk control strategies aim to either reduce or eliminate the likelihood of the threat occurrence, and to limit the impact of the risk factor. In our risk management framework, risk management can be viewed as two parts: risk identification and risk control. The former can reduce the occurrence probability of the risk factor through discovering risk timely, and the latter can control the negative impact on risk factors in sub-process products effectively. Risk checklists contain a set of generic risk items with brief descriptions that help identify possible sources of risk and develop awareness of the specific risks associated with a project. The expectation is that risk checklists could help software practitioners identify more risks and make better continuation decisions.

5. Trustworthy Software Process Risk Identification-Based on BNs

In this section, we introduce BN and risk correlation analysis into the software process model. Based on BN parameter learning and influence strength measurement, the process risk identification approach is proposed.

5.1. *Bayesian networks*

BN, also known as Bayesian belief networks (BBN), belongs to the family of probabilistic graphical models (GMs). A BN denoted as B is an directed acyclic graph (DAG) that represents a joint probability distribution (JPD) over a set of random variables V, in which the nodes represent variables and the edges express the dependencies between variables.[42] So, the network can be defined by a pair $B = \langle G, \Theta \rangle$, where G is the DAG whose nodes $X_1, X_2, \ldots, X_n$ represent random variables, and whose edges represent the direct dependencies between variables. Based on independent assumption, each node X_i in graph G is dependent of its parents in G. So, the graph G encodes the Markov assumption.[43] Θ represents the set of BN parameters, which contains the parameters $\theta_{x_i|\pi} = P_B(x_i|\pi_i)$ for each realization x_i of node X_i conditioned on π_i, where $\pi_i = \text{parent}(X_i)$.

Consequently, the network B can be defined as a unique JPD over variables V:

$$P_B(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P_B(x_i|\pi_i) = \prod_{i=1}^{n} \theta_{X_i|\pi_i}. \tag{1}$$

Based on BN, we cannot only capture the relationships between our uncertain beliefs in the propositions, but also find updated beliefs for each of the other variables by using BN inference algorithms if we learn the truth value of one or more of the propositions.[44] Each of these relationships can then be independently quantified using a sub-model suitable for the type and scale of information available.

In next subsection, we discuss the BN models in the context of process risk correlation. We elaborate that the BN models are suitable models for the process risk analysis and measurement.

5.2. *Risk correlation analysis during software process*

The risk correlation during software process can be classified into risk transfer and accumulation. Risk accumulation refers to the risk A of a sub-process would be cumulated and enlarged further in the next sub-process without risk control. Moreover, risk transfer refers that the risk A (or risk A and other risk factors) of a sub-process causes the occurrence of risk B in the next sub-process (See Fig. 1).

BN is a useful tool to integrate various kinds of information and in particular to study the risk's transfer and accumulation. DAG models the set of relationships among the risk factors; each of a DAG's nodes represents a risk factor and the arcs are the causal or influential links between the risk factors. A set of conditional probability functions associated with each node model the uncertain relationship between the risk factor and its parents. The cause-and-effect assumption of BN allows a complex causal risk chain linking actions to outcomes that would be factored into conditional relationships. So these relationships among the risk factors during software process can be constructed as a risk network like typical BN. The influence from one risk factor to another in the next sub-process can be described as the BN's conditional probability. Thus we can use BN to model the process risk correlation with trustworthy software process.

A BN encodes the probability distribution of a set of random variables by specifying a set of conditional independence assumptions together with a set of relationships among these variables and their related joint probabilities. Instead of deterministic results typically generated by learning, BN can yield a set of probabilistic results at each process. It is more appropriate for identifying and controlling risks at different stages of software process in management practice. For example, a simple 2-to-1 risk transfer can be represented as Fig. 2. In Fig. 2, through BN

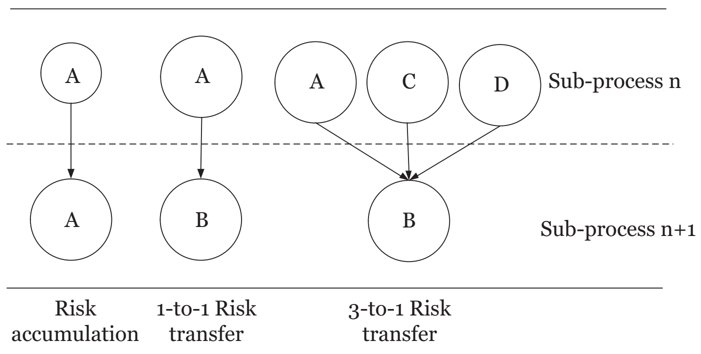

Fig. 1. Two genres of risk correlation presentation.

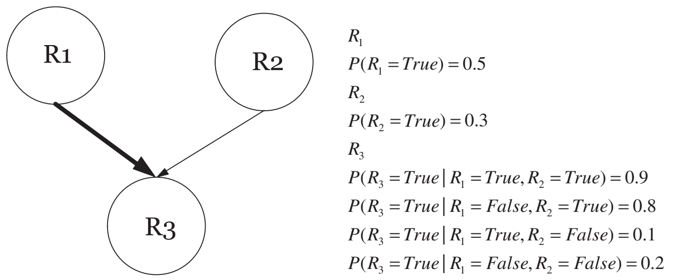
R1
R2
R3
R_1
$P(R_1 = True) = 0.5$
R_2
$P(R_2 = True) = 0.3$
R_3
$P(R_3 = True \mid R_1 = True, R_2 = True) = 0.9$
$P(R_3 = True \mid R_1 = False, R_2 = True) = 0.8$
$P(R_3 = True \mid R_1 = True, R_2 = False) = 0.1$
$P(R_3 = True \mid R_1 = False, R_2 = False) = 0.2$

Fig. 2. 2-to-1 risk transfer expressed by BN.

learning, the conditional probability between every two adjacent risk could be calculated. For example, $P(R_3 = \text{True}|R_1 = \text{True}, R_2 = \text{True}) = 0.9$ represents the occurrence probability of R_3 is 0.9 if R_1 and R_2 occur simultaneously. However, the conditional probability can't provide which one (R_1 and R_2) has larger influence on R_3. As is known, controlling the main risk source would be helpful to improve the performance of risk management under limited risk control resources. Therefore, we should analyze the influence strength from each parent node to its child nodes.

As shown in Fig. 1, the influence strength from A to B can be defined as the mutual information of (A, B) conditioned on all the other parents of B, as given in formula (2).[45] It describes the level how the parent nodes effect the child one at different parent node state.

$$\text{IS}(A, B) = \max_{b,a,c}(P(B > b_0|a, c) - P(B > b_0|a_0, c)), \tag{2}$$

where a, b are the states of nodes A and B and c is the state combination of all the other parent nodes except node B. Furthermore, a_0, b_0 and c_0 represent the default states of their corresponding nodes. In risk events, they can be considered as the state that the corresponding risk event doesn't occur.[46] $A > a_0$, $B > b_0$ and $C > c_0$ mean the corresponding risk event occurs. For example, in Fig. 2, influence strengths of R_1 to R_3 and R_2 to R_3 can be calculated as:

$$\text{IS}(R_1, R_3) = \max_{b,a,c}(P(R_3 = \text{True}) - P(R_3 = \text{True}|R_2 = \text{False})) = 0.9 - 0.1 = 0.8,$$
$$\text{IS}(R_2, R_3) = \max_{b,a,c}(P(R_3 = \text{True}) - P(R_3 = \text{True}|R_1 = \text{False})) = 0.9 - 0.2 = 0.7.$$

After visualization as Fig. 2, we can find the influence strength of R_1 to R_3 is stronger than R_2 to R_3. Generally, the influence strength value would be non-negative from the perspective of risk aversion. Therefore, this is a good understanding approach for measuring the risk network correlation.

Furthermore, we attempt to evaluate the risk importance (or impact) of each parent risk node if it occurs. A parent risk node may have several child risk nodes. Suppose the ith parent risk node in jth the sub-process is denoted as R_{ij}, where $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m-1\}$, n and m represent the number of risk categories and project sub-processes, respectively. The child risk nodes of R_{ij} are denoted as

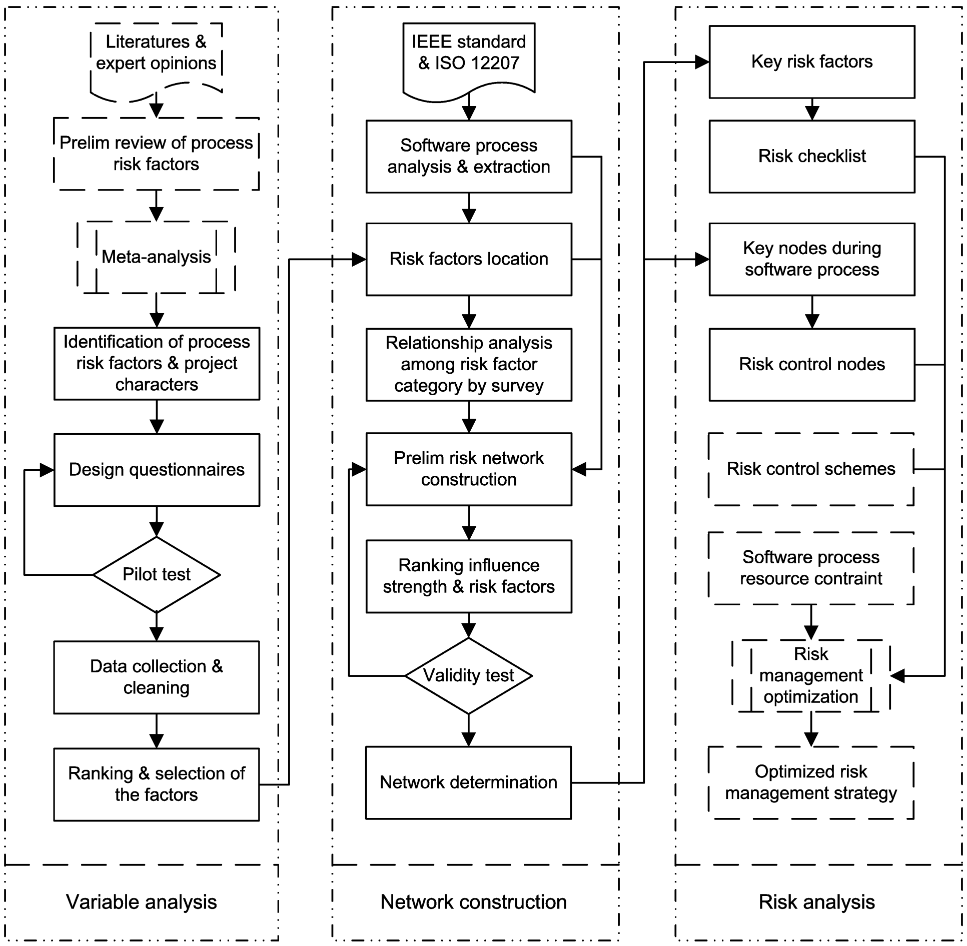
Literatures & expert opinions
Prelim review of process risk factors
Meta-analysis
Identification of process risk factors & project characters
Design questionnaires
Pilot test
Data collection & cleaning
Ranking & selection of the factors
Variable analysis
IEEE standard & ISO 12207
Software process analysis & extraction
Risk factors location
Relationship analysis among risk factor category by survey
Prelim risk network construction
Ranking influence strength & risk factors
Validity test
Network determination
Network construction
Key risk factors
Risk checklist
Key nodes during software process
Risk control nodes
Risk control schemes
Software process resource contraint
Risk management optimization
Optimized risk management strategy
Risk analysis

Fig. 3. The research framework of risk identification for trustworthy software.

$R_{i',j+1}^{\text{child}-i}$, where $i' \in S_i^{\text{child}}$ and S_i^{child} is the child risk node set of R_{ij}. According to formula (2), we can obtain a set of influence strength values between the parent risk node R_{ij} and its child risk nodes $R_{i',j+1}^{\text{child}-i}$ and they can be represented as IS $(R_{ij}, R_{i',j+1}^{\text{child}-i})$, where $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m-1\}$ and $i' \in S_i^{\text{child}}$. Apparently, the influence strength value can provide the influence strength between a pair of the parent risk node and its child risk node, but it cannot show the risk importance of the parent risk node intuitively. To provide a simple evaluation index, we first sum these influence strength values together, which can be represented as $\text{IV}_{ij} = \sum_{i'} \text{IS}(R_{ij}, R_{i',j+1}^{\text{child}-i})$ mathematically. Then we normalize IV_{ij} as the impact value of R_{ij}. The impact value can be seen as the measurement of the risk importance. Consequently, based on the ranking of impact values, we can get a key risk checklist.

In general, through BN learning, all conditional probabilities between every two adjacent risk could be calculated. All influence strength values between parent risk nodes and their child risk nodes can be obtained using formula (2). Based on the summation and normalization of influence strength values, we can get the impact value of a parent risk node. As a result, the risk importance can be ranked for risk checklist according to the impact values, which can be helpful to improve the performance of risk management.

5.3. *Research framework for risk identification*

Software risk identification is the first and critical activity adopted in software risk management process. For a reasonable and practical risk checklist we define our theoretical research model, which provides insights into checklist construction. As shown in Fig. 3, it includes three phases: variable analysis, network construction and risk analysis.

In the first phase (variable analysis), through review of the literatures and expert opinions in industrial standards, we can get a crude risk checklist of software process through the meta-analysis. This phase is the basis of our research. Then we design the questionnaires to obtain the risk knowledge of practitioners. After data collection, we can get a preliminary process risk checklist.

In the second phase (network construction) and the third phase (risk analysis), which are the focus of this paper, we construct the network and identify the important risk factors using BN approach. After parameter learning based on BN, we can calculate the risk influence strength. Key risk nodes and risk checklist can be acquired.

At last, based on multi-criteria optimization approach, the optimized software process risk management plan and risk control strategy set against different risk factors during different sub-processes could be obtained for risk management resource allocation.

6. Case Study

In this section, we obtain practitioners' understanding about process risk factors by questionnaire and construct the BN for risk analysis.

6.1. *Overview of survey data*

The meta-analysis approach and unstructured interviews are conducted with an expert group. This group is composed of more than 20 software project managers and software process management researchers. Then 52 risk factors affecting the software trustworthiness are extracted from various risk factors by these experts.

To obtain project managers' understanding about software process risk for different industry software development environments and different software companies, we have designed the questionnaire for discovering the risk occurrence probability and its impact to trustworthiness. Because lacking of clarity in defining trustworthy software may lead to misleading conclusions from the questionnaires, the definition of trustworthy software was explained to all respondents at the start of questionnaires clearly. In the questionnaires, software trustworthiness is defined as the level of software product meeting the requirements of users. The demands include software component, application, system and network.

The questionnaire is composed of the following four parts:

Part A. *The basic personnel information and enterprise backgrounds of the respondents.*

Part B. *The performance of the last software project process.* It includes software performance, difficulty level to maintain the software, function completion degree, budget utilization and the gap with the original planning schedule.

Part C. *Occurrence and influence on software trustworthiness of the 52 risk factors during six software processes (Planning process; Requirement process; Design process; Development process; Testing process; Implementation process).* This part aims to help us use the BN model to learn the parameters of the network.

Part D. *The basic influence relationships among the factors during the whole software process.* These data are gathered to identify cause–effect relationships among factors to construct the initial BN.

The questionnaire consists of six risk categories (Requirement risk; Project management risk; Technical risk; Developer risk; User risk; Organization and environment risk) with 58 questions. The question form of the questionnaire is designed as shown in Table 1. Each risk category contains several questions and these 52 risk factors information are grouped into the six risk categories. Plan, Spec, Design, Build, Test and Imple represent Planning process, Requirement process, Design process, Development process, Testing process and Implementation process, respectively. The respondent fills in the questionnaire according to his/her experience. During six sub-processes, GeNIe (http://genie.sis.pitt.edu/) and Elvira (http://www.ia.uned.es/∼elvira/index-en.html) are used to develop the BN-based model based on the six risk categories and the cause–effect relationships of risk factors.

Table 1. Part of the questionnaire.

I. Requirement risk	Does the risk exist?		If it exists, which sub-process does it happen?					
			Plan	Spec	Design	Build	Test	Imple
1. Unclear system requirements	☐ Yes	☐ No						
		......						
		......						
III. Project management risk	**Does the risk exist?**		**If it exists, which sub-process does it happen?**					
			Plan	Spec	Design	Build	Test	Imple
20. Excessive schedule pressure/Long schedule	☐ Yes	☐ No						
		......						

Finally, the occurrence probability and influence of process risk factors can be inferred by applying the BN model.

A total of 100 copies of the questionnaire were distributed to managers involved in software projects in five large software enterprises in China. A total of 93 responses were received. Companies of 86.2% of respondents have passed the CMMI Levels 4 and 5, and all companies have more than 500 employees. 85.6% of respondents are project managers experienced in software development.

To verify the reliability of the survey data, we applied Cronbach's alpha value for all risk factors in the questionnaire.[47] The result shows that almost all variables' Cronbach's alpha values are higher than 0.85 (all variables' Cronbach's alpha values are higher than 0.7), which indicates that the survey data can satisfy the requirement for BN construction.

Besides, we can get the distribution of risk categories in each individual sub-process according the questionnaire data. Suppose a risk category consists of m risk factors. k_i is the number of respondents who believe the ith risk factor will happen in the Planning process. n is the total number of respondents. As a result, the occurrence probability of the risk category in the Planning process would be $\sum_{i=1}^{m} k_i/n$. As shown in Fig. 4, we can find that the occurrence probability of each risk category in different sub-processes might be different. Also, the distributions of the occurrence probability of different risk categories are distinguishing. These indicate that it is necessary to consider the process in software risk analysis.

6.2. *Bayesian network construction*

The software process is the origin of the software. The trustworthiness of software product would be affected by all the risk factors during the software process. Because of risk transfer and accumulation mechanism, six categories or 52 types of risk factors in each individual sub-process transfer and accumulate continuously by affecting the occurrence probability of risk factors in the next sub-process.

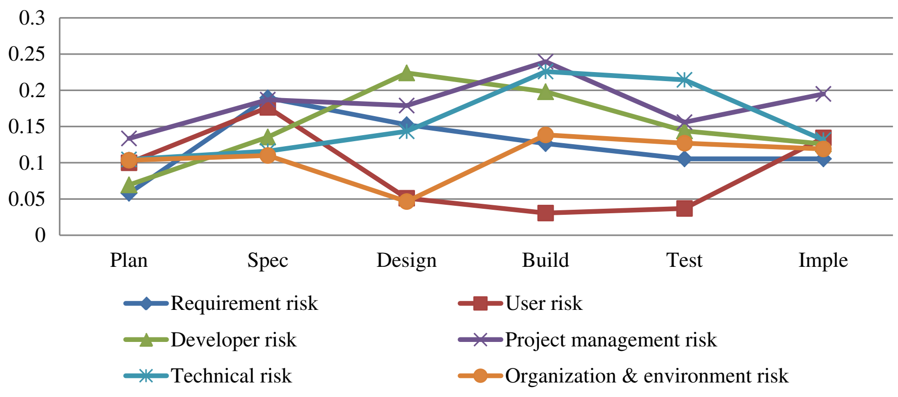

Fig. 4. Distributions of the occurrence probability for different risk categories.

Every risk occurrence during the software process could make influence on the final product trustworthiness on varying levels. Furthermore, software trustworthiness includes several attributes. Different deliverables or services of sub-processes would influence the trustworthiness attribute values of the software product differently. These differences on influence levels can be denoted by a risk impact matrix which is determined by experts according to the demand of product. Without loss of generality, in this paper the risk impact matrix is defined as a matrix which has the same element, that is, each sub-product has the equivalent influence on every trustworthiness attribute. Therefore, the final trustworthiness value equals to the sum of all the risk impact as shown in Fig. 5.

Based on the risk factors during the whole software process and the cause-and-effect relationships among risk factors, a conceptual BN model was constructed using GiNIe (See Fig. 6). The oval nodes of the network indicate the process risk factors in different sub-processes. All the arcs between every two oval nodes at adjacent sub-processes indicate the influence of nodes at arc tails on the nodes at heads. Risk factors in each sub-process could threaten its sub-product's quality which affects the final trustworthiness. Besides, the rectangle nodes indicate that risk factors in each individual row belong to the same sub-process. The hexagon node represents the software trustworthiness, which is influenced by the risk factors in each individual sub-process. The rectangle and hexagon nodes are used to make the explanation. It should be noteworthy that their shape is meaningless. In addition, the green nodes named "Sub-Product" are not variables, but the explanation to show that the quality of each sub-product is affected by all risk factors in the corresponding sub-process.

Then the investigated data are imported to the existing network (structure already defined) to learn parameters. A mapping between the nodes defined in the network and risk occurrence defined in the data set is created. Furthermore, the

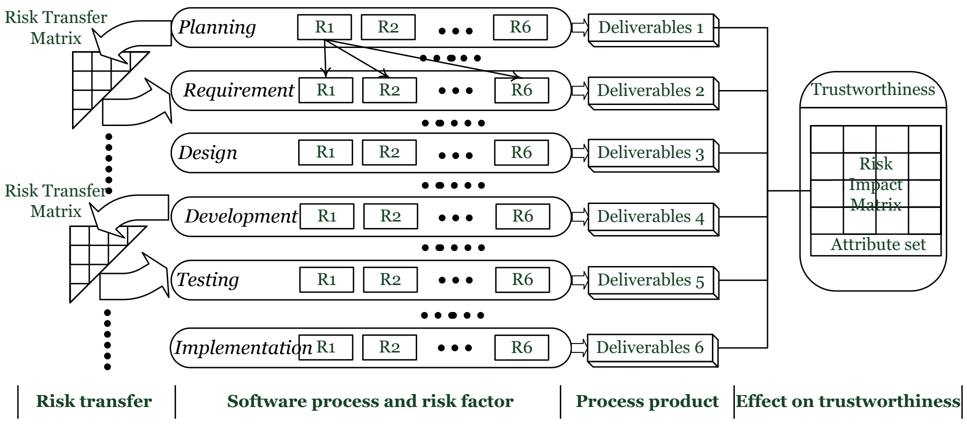

Fig. 5. The framework of process, risk, deliverables and trustworthiness.

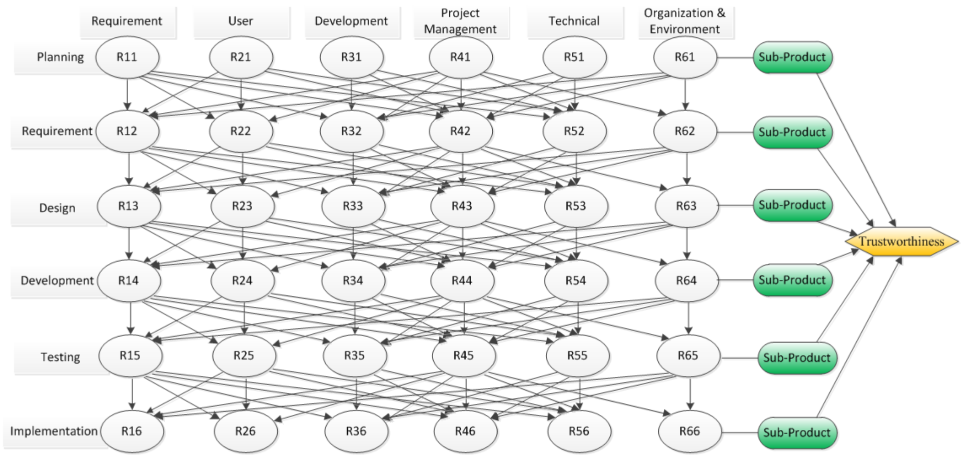

Fig. 6. BN-based model for inferring the occurrence probability and influence of process risk factors.

mapping of the states of those variables is determined as well. Each risk has two statements: True or False. True represents this risk occurs in its sub-process. False represents nonoccurrence.

In addition to the DAG structure, which is often considered as the "qualitative" part of the model, one needs to specify the "quantitative" parameters of the model. The parameters are described in a manner which is consistent with a Markov property, where the conditional probability distribution (CPD) at each node depends only on its parents. For discrete random variables, this conditional probability is often represented by a table, listing the local probability that a child node takes on each of the feasible values — for each combination of its parent node statements. The joint distribution of a collection of variables can be determined uniquely by these local conditional probability tables (CPTs).

To preserve the information of the survey data in the great extent, we choose the expectation maximization (EM) algorithm as the parameter learning approach to find a locally optimal maximum likelihood estimation of the parameters in GeNIe. Using the BN parameter learning approach, the CPTs are calculated. Then the BN with parameters can be obtained as shown in Fig. 7.

6.3. *Risk analysis*

According to the definition of the influence strength, the impact intensity of the risks can be calculated between the different sub-processes. After GiNie's visualization, the influence strength of the Bayesian risk correlation with network diagrams can be shown in Fig. 7. The numbers behind "True" and "False" in the rectangle denote the occurrence and nonoccurrence probabilities of the sub-process risks respectively. And the rectangles represent the risk R_{ij} like Fig. 6.

Each risk factor has its specific occurrence probabilities in different sub-processes. The width of the arrow represents the influence strength from the parent risk. These

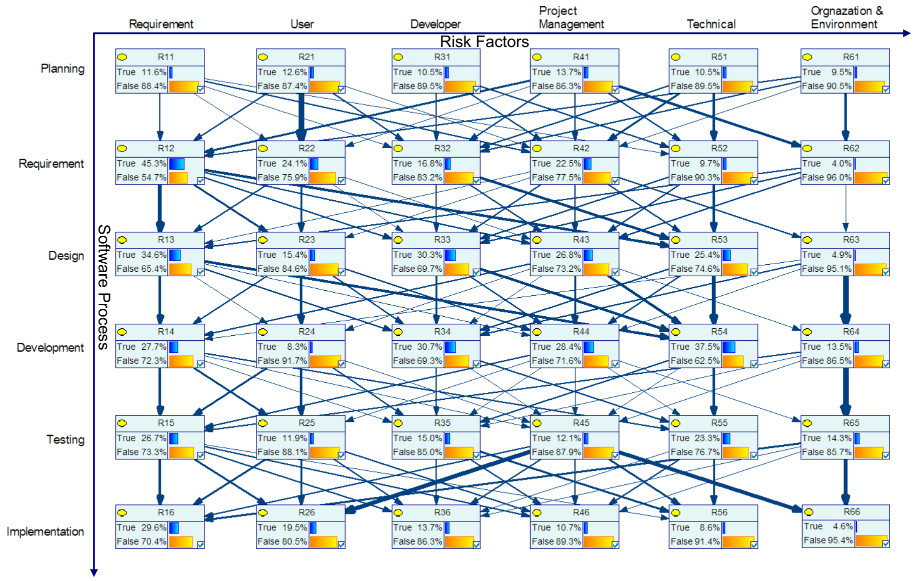
Requirement
User
Developer
Project Management
Technical
Orgnazation & Environment
Risk Factors
Software Process
Planning
Requirement
Design
Development
Testing
Implementation
R11 True 11.6% False 88.4%
R21 True 12.6% False 87.4%
R31 True 10.5% False 89.5%
R41 True 13.7% False 86.3%
R51 True 10.5% False 89.5%
R61 True 9.5% False 90.5%
R12 True 45.3% False 54.7%
R22 True 24.1% False 75.9%
R32 True 16.8% False 83.2%
R42 True 22.5% False 77.5%
R52 True 9.7% False 90.3%
R62 True 4.0% False 96.0%
R13 True 34.6% False 65.4%
R23 True 15.4% False 84.6%
R33 True 30.3% False 69.7%
R43 True 26.8% False 73.2%
R53 True 25.4% False 74.6%
R63 True 4.9% False 95.1%
R14 True 27.7% False 72.3%
R24 True 8.3% False 91.7%
R34 True 30.7% False 69.3%
R44 True 28.4% False 71.6%
R54 True 37.5% False 62.5%
R64 True 13.5% False 86.5%
R15 True 26.7% False 73.3%
R25 True 11.9% False 88.1%
R35 True 15.0% False 85.0%
R45 True 12.1% False 87.9%
R55 True 23.3% False 76.7%
R65 True 14.3% False 85.7%
R16 True 29.6% False 70.4%
R26 True 19.5% False 80.5%
R36 True 13.7% False 86.3%
R46 True 10.7% False 89.3%
R56 True 8.6% False 91.4%
R66 True 4.6% False 95.4%

Fig. 7. The probabilities and influence strength for risk factors based on the BN model.

wide arrow lines point out the sub-processes which risk management should focus on. The risks having the large probability or have relatively significant impact on the next process should be paid more attention in process management.

We can get on the six categories of risks in the entire software process statistical description in bubble chart as Fig. 8. The bubble represents the impact degree of the sub-process risk factor on the software trustworthiness. The bigger bubble is, the larger impact degree of the sub-process risk factor on the software trustworthiness would be. We can easily find that requirement and project management risk are the most important risks in the whole software process. Especially, requirement risks have a significant impact on the trustworthiness of software products except the planning sub-processes. This is consistent with most of the risk checklist. In addition, we also find that project management risks play an important role throughout the process, especially in the beginning and the final sub-processes. Organization and environment risks have a stronger effect in design, development and testing periods than the beginning. However, user risks do not just exist and influence the product at the beginning like imagination. They are also important risk factors during the whole software process. In comparison, technical and developer risks are not critical risks to the trustworthy software product in the view of China's high CMMI level companies. The introduction of relatively few new software technologies should be the main reason for this outcome. In addition, the implementation phase is the last phase, which indicates that it does not exist the risk transfer and accumulation to the next sub-process, thus it is not depicted in this figure.

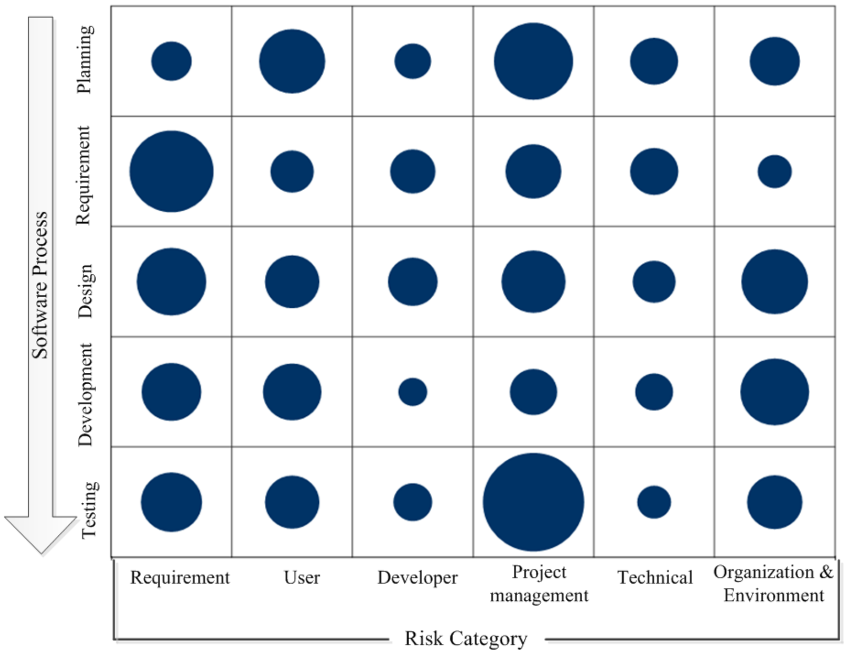

Fig. 8. The risks' influence strength during the software sub-processes.

Furthermore, we can construct the BN by using 52 types of risks instead of the six categories of risks. Similarly, through BN learning, we can get the probability of every risk factor. The top 20 process risk factors ordered by the impact can be shown in Table 2 after normalization. This table can be used in the software process risk management directly after customization. In our checklist, every risk has its probability and impact value in each individual sub-process. According to this checklist, requirement risk of requirement analysis sub-process; technical, developer risk and project management risk of development sub-process and project management risk of development sub-process should be paid more attention in risk management. For example, *unclear system requirement, client's lack of knowledge about the software requirements, incorrect system requirement, continually changing system requirements, undefined project success criteria* and *conflicting system requirements* are the most important risks during requirement analysis process. In addition, Han and Huang[48] have provided the top 10 software risks, which are *Continually changing system requirements, System requirements not adequately identified, Unclear system requirements, Lack of an effective project management methodology, Incorrect system requirements, Poor project planning, Inadequate estimation of required resources, Project involved the use of new technology, Project progress not monitored closely enough, Corporate politics with negative effect on project*, respectively. We can find that most of these risks are also included in our risk checklist. Both of us consider that requirement risk and project management risk are the two important

Table 2. Top 20 software process risk factors ordered by the impact.

Rank	Risk list	Risk category	Sub-process	Probability (%)	Impact
1	Unclear system requirements	Requirement risk	Requirement analysis	26	0.78
2	Client's lack of knowledge about the software requirements	Requirement risk	Requirement analysis	28	0.75
3	Incorrect system requirements	Requirement risk	Requirement analysis	23	0.68
4	Continually changing system requirements	Requirement risk	Requirement analysis	21	0.60
5	Introduction of new technology	Technical risk	Development	20	0.48
6	Lack of Required Knowledge/Skills in the Project Personnel	Developer risk	Development	16	0.42
7	Excessive schedule pressure/Long schedule	Project management risk	Development	16	0.37
8	Failure to manage end user expectations	Requirement risk	Requirement analysis	13	0.36
9	Dependence on a few key people	Developer risk	Development	13	0.34
10	Inexperience with project's platform/environment/methods	Technical risk	Development	15	0.33
11	Undefined project success criteria	Requirement risk	Requirement analysis	12	0.33
12	Need to integrate/interface with other systems	Technical risk	Development	14	0.32
13	Conflicting system requirements	Requirement risk	Requirement analysis	12	0.32
14	Instability and lack of continuity in project staffing	Developer risk	Development	13	0.28
15	Project leader's experience	Project management risk	Design	11	0.28
16	Extent of changes in the project	Project management risk	Design	11	0.27
17	Too many staff and poor cooperation	Developer risk	Development	11	0.26
18	Ineffective communication	Project management risk	Design	12	0.25
19	Lack of organizational maturity	Developer risk	Design	11	0.25
20	Lack of staff commitment, low morale	Developer risk	Development	12	0.24

Table 3. The probability and impact of "*Introduction of new technology*" in each sub-process.

Sub-process	Planning	Requirement analysis	Design	Development	Testing	Implementation
Probability	7.1%	6.7%	15.2%	20.5%	13.3%	9.6%
Impact	0.12	0.08	0.21	0.48	0.16	0.19

risk categories that are more likely to generate the risk factors with high negative impacts. Besides of the two important risk categories, we also consider that technical risk and developer risk are another important risk categories for concern. Therefore, our risk checklist is more comprehensively to some extent, which can give a more effective decision support for project managers.

The same risk factor would occur in different sub-processes and have the different probabilities and impacts on the trustworthiness. For example, the risk factor, "*Introduction of new technology*", is an important risk factor in development sub-process. And it also has the large probability and impact in design sub-process as illustrated in Table 3. To achieve a better risk response effect, more risk management resources should be invested in these two sub-processes other than other sub-processes.

7. Conclusions

In this paper, we propose how BN is applied to quantify the occurrence probability of software process risk factors and the influence strength among the process risk factors. The information about 52 factors was obtained through a questionnaire survey of 93 project managers in five software enterprises in China. In Chinese IT software industry, software process data, especially risk management data, are very scarce and difficult to obtain. Therefore, it is very important to get these data for process risk management research in China. Based on these data, we could provide a novel process risk checklist for Chinese high CMMI level software companies. It is different from the previous simple checklist for the risk of the whole software process. Our checklist is more suitable for process risk management, which includes the probability and impact of each risk factor in each individual sub-process. For the purpose of "do the right thing, at a good time", it could help software risk managers to allocate limited risk management resources at the important sub-processes. And it can also offer good timings for process risk control to improve software process risk management. Based on the BN with parameters, at each phase of the software process, we can simulate or predict the risk during the next sub-processes using the data of risks in finished sub-processes, which would make risk management more effective.

The findings of this study can be used to provide the conditional probability tables of risk factors to software risk manager for the key risk factor identification and process risk management decision-making. Our approach is general and can be applied to a certain software project or some software enterprises with updated data.

Furthermore, we have several future directions to pursue. First, an extended conditional probability table should be constructed. The risk states in this paper only include TRUE or FALSE, and they can be extended more occurrence states for better and more explicit results. Second, for the various risks in each individual sub-process, the risk checklist should be expanded with specific risk control approach under different trustworthiness demands. It could make the checklist more practical and meaningful.

Acknowledgments

This research has been supported by grants from the National Natural Science Foundation of China (71425002, 71571179, 91218302, 71201156), Youth Innovation Promotion Association of the Chinese Academy of Sciences (2013112), and China Postdoctoral Science Foundation funded project (2015M571135).

References

1. I. Keivanloo and J. Rilling, Software trustworthiness 2.0: A semantic web enabled global source code analysis approach, *Journal of Systems and Software* **89** (2014) 33–50.
2. X. Zhong and X. Zhao, Method for software trustworthiness evaluation supporting dynamic and multiple attributes, *Computer Engineering and Science* **35**(6) (2013) 107–112.
3. Y. B. Zhong, Z. L. Liu and X. H. Yuan, A fuzzy trustworthiness system with probability presentation based on center-of-gravity method, *Annals of Data Science* **2**(3) (2015) 335–362.
4. G. Kou, Y. Shi and G. Dong, Data mining for software trustworthiness, *Information Sciences* **191** (2012) 1–2.
5. G. Kou, Y. Peng, Y. Shi and W. Wu, Classifier evaluation for software defect prediction, *Studies in Informatics and Control* **21**(2) (2012) 117–126.
6. Y. Peng, G. Kou, G. Wang, H. Wang and F. I. Ko, Empirical evaluation of classifiers for software risk management, *International Journal of Information Technology & Decision Making* **8**(4) (2009) 749–767.
7. L. Bernstein, Trustworthy software systems, *ACM SIGSOFT Software Engineering Notes* **30**(1) (2005) 4–5.
8. W. Hasselbring and R. Reussner, Toward trustworthy software systems, *Computer* **39**(4) (2006) 91–92.
9. K. Dejaeger, T. Verbraken and B. Baesens, Toward comprehensible software fault prediction models using bayesian network classifiers, *IEEE Transactions on Software Engineering* **39**(2) (2013) 237–257.
10. M. Staron, W. Meding and C. Nilsson, A framework for developing measurement systems and its industrial evaluation, *Information and Software Technology* **51**(4) (2009) 721–737.
11. M. Silic and A. Back, The influence of risk factors in decision-making process for open source software adoption, *International Journal of Information Technology & Decision Making* **15**(01) (2016) 151–185.
12. B. Boehm, A view of 20th and 21st century software engineering, in *Proc. 28th Int. Conf. Software Engineering* (Shanghai, China: ACM, 2006).
13. K. De Bakker, A. Boonstra and H. Wortmann, Does risk management contribute to IT project success? A meta-analysis of empirical evidence, *International Journal of Project Management* **28**(5) (2010) 493–503.

14. S. Liu and L. Wang, Understanding the impact of risks on performance in internal and outsourced information technology projects: The role of strategic importance, *International Journal of Project Management* **32**(8) (2014) 1494–1510.
15. N. Davis, W. Humphrey, S. T. Redwine, G. Zibulski and G. McGraw, Processes for producing secure software, *IEEE Security & Privacy* **2**(3) (2004) 18–25.
16. S. Du, M. Keil, L. Mathiassen, Y. Shen and A. Tiwana, Attention-shaping tools, expertise, and perceived control in IT project risk assessment, *Decision Support Systems* **43**(1) (2007) 269–283.
17. C.-F. Fan and Y.-C. Yu, BBN-based software project risk management, *Journal of Systems and Software* **73**(2) (2004) 193–203.
18. L. Wallace, M. Keil and 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.
19. J. Jiang and G. Klein, Risks to different aspects of system success, *Information Management* **36**(5) (1999) 263–271.
20. C. Li, S. Li and Y. Liu, Method of power supply mode selection for urban distribution network planning based on association rules, *Journal of Systems Science and Information* **3**(5) (2015) 421–433.
21. Z. Xu, B. Yang and 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.
22. H. Samadi, S. Nazari-Shirkouhi and A. Keramati, Identifying and analyzing risks and responses for risk management in information technology outsourcing projects under fuzzy environment, *International Journal of Information Technology & Decision Making* **13**(6) (2014) 1283–1323.
23. L. Wallace, M. Keil and A. Rai, Understanding software project risk: A cluster analysis, *Information Management* **42**(1) (2004) 115–125.
24. D. Neumann, An enhanced neural network technique for software risk analysis, *IEEE Transactions on Software Engineering* **28**(9) (2002) 904–912.
25. Y. Hu*et al.*, Software project risk analysis using Bayesian networks with causality constraints, *Decision Support Systems* **56** (2013) 439–449.
26. N. Feng, H. J. Wang and M. Li, A security risk analysis model for information systems: Causal relationships of risk factors and vulnerability propagation analysis, *Information Sciences* **256** (2014) 57–73.
27. D. Hanea and B. Ale, Risk of human fatality in building fires: A decision tool using Bayesian networks, *Fire Safety Journal* **44**(5) (2009) 704–710.
28. P. A. P. Ramírez and I. B. Utne, Use of dynamic Bayesian networks for life extension assessment of ageing systems, *Reliability Engineering & System Safety* **133** (2015) 119–136.
29. N. E. Fenton and M. Neil, Decision support software for probabilistic risk assessment using bayesian networks, *IEEE Software* **31**(2) (2014) 21–26.
30. E. J. Lauría and P. J. Duchessi, A Bayesian belief network for IT implementation decision support, *Decision Support Systems* **42**(3) (2006) 1573–1588.
31. E. Lauría and 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.
32. E. Lee, Y. Park and J. G. Shin, Large engineering project risk management using a Bayesian belief network, *Expert Systems with Applications* **36**(3) (2009) 5880–5887.
33. N. Fenton *et al.*, Predicting software defects in varying development lifecycles using Bayesian nets, *Information and Software Technology* **49**(1) (2007) 32–43.

34. A. C. V. de Melo and A. J. Sanchez, Software maintenance project delays prediction using Bayesian Networks, *Expert Systems with Applications* **34**(2) (2008) 908–919.
35. J. Li, M. Li, D. Wu and H. Song, An integrated risk measurement and optimization model for trustworthy software process management, *Information Sciences* **191** (2012) 47–60.
36. D. Zhang *et al.*, Incorporation of formal safety assessment and Bayesian network in navigational risk estimation of the Yangtze River, *Reliability Engineering & System Safety* **118** (2013) 93–105.
37. R. J. Kohl, Trustworthy Software, in *Proc. Center for National Software Studies Workshop on Trustworthy Software* (Monterey, California, 2004).
38. B. Boehm, Software risk management: Principles and practices, *IEEE Software* **8**(1) (1991) 32–41.
39. M. Keil *et al.*, An investigation of risk perception and risk propensity on the decision to continue a software development project, *Journal of Systems and Software* **53**(2) (2000) 145–157.
40. M. Keil, H. K. Lee and T. Deng, Understanding the most critical skills for managing IT projects: A Delphi study of IT project managers, *Information & Management* **50**(7) (2013) 398–414.
41. P. L. Bannerman, Risk and risk management in software projects: A reassessment, *Journal of Systems and Software* **81**(12) (2008) 2118–2133.
42. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann Series in Representation and Reasoning* (Morgan Kaufmann, San Mateo, CA, US, 1998), p. 552.
43. P. Baraldi *et al.*, Comparing the treatment of uncertainty in Bayesian networks and fuzzy expert systems used for a human reliability analysis application, *Reliability Engineering & System Safety* **138** (2015) 176–193.
44. V. J. Finn and D. N. Thomas, *Bayesian Networks and Decision Graphs* (Springer, Berlin, 2007).
45. C. Lacave and F. J. Díez, A review of explanation methods for Bayesian networks, *The Knowledge Engineering Review* **17**(2) (2002) 107–127.
46. A. Okutan and O. T. Yıldız, Software defect prediction using Bayesian networks, *Empirical Software Engineering* **19**(1) (2014) 154–181.
47. J. Reynaldo and A. Santos, Cronbach's alpha: A tool for assessing the reliability of scales, *Journal of Extension* **37** (1999) 2–3.
48. W. Han and S. Huang, An empirical analysis of risk components and performance on software projects, *Journal of Systems and Software* **80**(1) (2007) 42–50.