

Software Change Project Cost Estimation – A Bayesian Network and a Method for Expert Elicitation

Robert Lagerström, Pontus Johnson, David Höök, Johan König
Industrial information and control systems
The Royal Institute of Technology, Stockholm, Sweden
{robertl, pj101, davidh, johank}@ics.kth.se

Abstract

Business environments today progress and change rapidly. Most business processes are supported by information systems and since the business processes change, the information systems need to be modified as well. An essential issue with today's software systems is that many of them are interconnected, thus a change to one system may cause a ripple effect among other systems. Also, numerous systems have been developed and modified during many years and to make further changes to them requires a lot of effort from the organization. This paper proposes a Bayesian network for analysis of software change projects. The network contains a set of variables affecting the cost of making software changes both on architectural and component level. An expert survey was conducted in order to find the strength of the effect between all the related variables. The survey results and the expert elicitation method used are described. Two case studies including ten software change projects have been evaluated with the resulting network. These predictions are compared with the actual cost outcome to show the applicability of the network.

Index Terms: Bayesian networks, software change projects, change cost, expert survey, expert elicitation

1. Introduction

Business environments today progress and change rapidly to keep up with evolving markets. Most business processes are supported by information systems and since the business processes change, the information systems need to be modified as well in order to continue supporting the processes. These modifications can differ from adding a functional requirement in one system to implementing a service oriented architecture for the whole enterprise.

An essential issue with today's software systems is that many of them are interconnected, thus a modification to one system may cause a ripple effect among other systems. Also, numerous systems have been developed and modified during many years. To make further changes to these systems might require a lot of effort from the organization, for example due to a large amount of previous modifications implemented ad hoc. Problems like these pose questions for IT decision makers such as: Is there enough documentation describing the systems, and has the documentation been updated correctly after each modification? Is the source code easy to understand? Which systems are interconnected?

The activities of modifying enterprise systems are typically executed in projects and IT decision makers often find it difficult to predict and plan their change projects. Thus, a large proportion of the projects with the purpose of modifying a software system environment fail. That is, the projects tend to take longer time and cost more than expected. This can often occur due to lack of information about the systems. Therefore, it would be useful for IT decision makers to gather more information in a structured manner and use this information to analyze how much effort a certain modification to an enterprise information system would require.

Bayesian networks can be used as an approach to analyze enterprise information systems. It is an approach relying on models containing variables and their effect on each other. Thus, instead of starting software change projects using trial and error, a Bayesian network is proposed to predict the behavior and effects of modifications to enterprise systems. The network allows reasoning about the consequences of various scenarios and thereby support decision making.

The purpose of this paper is to present a Bayesian network as a prediction model for software change project cost analysis, an expert elicitation method, and expert data gathered in surveys and workshops. The

network is intended to be employed in decision situations when prioritizing, choosing, and planning change projects. The results from the network enable the decision maker to: (1) make predictions of the change cost for different software projects and (2) retrieve information for risk analysis in the change projects. In order for the network to be able to predict change projects well, the network is based on experience from previously implemented change projects. This experience data has been gathered in workshops with industry experts and in surveys with academic experts.

The remainder of the paper is delineated as follows; section 2 introduces Bayesian networks. Section 3 presents the related work on the topic of previously proposed software change cost analysis methods, as well as related Bayesian learning methods and expert elicitation approaches. The following section, section 4 presents the software change project cost network. Next, in section 5 approaches for setting conditional probabilities of Bayesian networks are discussed, the expert data collection and results are presented, and the expert elicitation method for the cost network with workshop and survey data is presented in the subsection 5.1. Section 6 shows the applicability of the network by presenting two case studies with ten change projects. Finally, section 7 summarizes the paper.

2. Bayesian networks

A Bayesian network, $B=(G, P)$, is described as a representation of a joint probability distribution, where $G=(V, E)$ is a directed acyclic graph consisting of vertices, V , and edges, E [1]. The vertices denote a domain of random variables X_1, \dots, X_n , also denoted chance nodes. Each chance node, X_i , may take on a value x_i from the finite domain $Val(X_i)$. The edges denote causal dependencies between the nodes, i.e. how the nodes relate to each other. The second component, P , of the network B , describes a conditional probability distribution for each chance node, $P(X_i)$, given its parents $Pa(X_i)$ in G . It is possible to write the joint probability distribution of the domain X_1, \dots, X_n using the chain rule of probability, in the product form:

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i | Pa(X_i))$$

In order to specify the joint distribution, the respective conditional probabilities that appear in the product form must be found. The second component P describes distributions for each possible value x_i of X_i ,

and $pa(X_i)$ of $Pa(X_i)$, where $pa(X_i)$ is the set of values of $Pa(x_i)$. These conditional probabilities are represented in matrices, here on called conditional probability matrices (CPMs).

If the probabilities of the source variables are known, it is possible to infer a value for the target variable using the law of total probability,

$$P(X_1) = \sum_i P(X_1 | Pa(X_i)) P(Pa(X_i)).$$

Also, using Bayes' rule,

$$P(X_1 | Pa(X_1)) = \frac{P(Pa(x_1) | X_1) P(X_1)}{P(Pa(x_1))}$$

makes it possible to calculate the values of source variables based on the probabilities of a target variables.

A Bayesian network example is presented in Figure 1 and the corresponding conditional probability matrix is presented in Table 1.

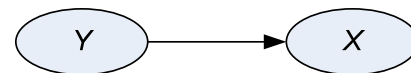


Figure 1. A Bayesian network example, where variable Y affects variable X.

In the example the probability matrix represents the probabilities of the variable X to be x_1, x_2 , or x_3 , if a variable Y is y_1, y_2 , or y_3 . As can be seen in the figure the value of variable X would be x_1 with a probability $P(x_1|y_2)$ if the value of variable Y is y_2 .

Table 1. The corresponding conditional probability matrix for the example network presented in Figure 1.

Y		y_1	y_2	y_3
X	x_1	$P(x_1 y_1)$	$P(x_1 y_2)$	$P(x_1 y_3)$
	x_2	$P(x_2 y_1)$	$P(x_2 y_2)$	$P(x_2 y_3)$
	x_3	$P(x_3 y_1)$	$P(x_3 y_2)$	$P(x_3 y_3)$

For more comprehensive treatments on Bayesian networks see [2, 3].

3. Related works

Several methods for analyzing software change costs and the related topic of maintainability (sometimes also called modifiability) have been suggested, for example the COConstructive COSt MOdel (COCOMO) for software maintenance cost estimation [4], the Definition and Taxonomy for Software

Maintainability [5], the maintainability part of the Quality Model proposed by ISO/IEC [6], the maintainability software metrics suggested by Fenton and Pfleeger [7], and the Software Architecture Analysis Method (SAAM) concerning modifiability [8].

Some of these methods focus on the cost of software development and not software modifications. Other methods have their centre of attention on component level development and change. Few of these methods focus on the change project cost for enterprise information systems management, i.e., taking the surrounding environment and the coupling between systems etc into consideration.

In the field of Bayesian networks learning there are numerous methods suggested, some of the more famous ones are, the Expectation Maximization (EM) [9], B-Course [10], Path Condition (PC), and the Necessary Path Condition (NPC) algorithms [11].

The four mentioned methods all require statistical data. In this case for instance, it would mean studying numerous change projects. Collecting the type of statistical data in the amount necessary for these methods is very time consuming. It would require data collection from hundreds of change projects.

A common expert experience based technique is the expert elicitation approach and it focuses on letting experts assess and set probabilities into Bayesian networks [12, 13]. However, most experts are unfamiliar with concepts such as Bayesian networks and conditional probabilities, thus an introductory phase is most often necessary. Further discussions on the complexity of Bayesian learning methods and expert elicitation approaches can be found in [14, 15].

Since the available methods for learning Bayesian networks do not consider expert opinions and since no appropriate method for expert elicitation has been found, this paper introduces a newly developed method. The proposed method takes expert opinions into consideration without the need of introducing the experts to the concepts of conditional probabilities and Bayesian networks.

4. The software change project cost network

This section presents a Bayesian network for software change project cost prediction, c.f. Figure 2. The network is to be employed when analyzing the cost of software change projects, i.e. to predict the number of man-hours needed to make a certain modification. An IT decision maker can also benefit from using the network when doing change project risk analysis.

The presented network is influenced by and based on earlier suggested cost and maintainability frameworks, scientific papers, and books, such as the work by Boehm [4], Oman et al. [5], ISO [6], Fenton [7], Bass et al. [8], Brown et al. [16], Broy et al. [17], Bengtsson et al. [18], Matinlassi et al. [19], Aggarwal et al. [20], Granja-Alvarez et al. [21], Chan et al. [22], Grubb et al. [23], Smith [24], and Pigoski [25].

Instead of focusing on a single software system or component, as many have done before, this network intends to support software change cost of enterprise systems, i.e. enterprise wide systems of systems. In modern system architectures, i.e. systems that are tightly coupled, there are new problems occurring, problems that need to be taken into account when planning for modifications.

The network is divided into two parts; one with focus on architectural modifications and the other on component level modifications. Both parts have variables considering the people, documentation, tools, and infrastructure involved. The main difference is that the architectural side reflects on systems and architects while the component side reflects on developers and components. The network is presented in Figure 2 and the usage of the network and previous versions have been published in [26, 27].

The Bayesian network focuses on the software systems and the surrounding environment involved in or affected by the modifications implemented in a change project. A change project could typically be a project adding functionality to systems already in use or integrating a newly bought system with other systems at the enterprise. The main variable is the *change project cost*. The cost of a change project is measured as the number of man-hours needed to make the modification. As can be seen in Figure 2 the relations connected to the change project cost variable is different from all the other relations. It is indicated as a dashed line with two plus (+) signs. This originates in the fact that these relations are defined as the sum of the costs for architectural and component changes and thus not a part of the survey described in section 6.

All modifications related to software systems need to be formally managed in a controlled manner, including changes to be logged, assessed and authorized prior to implementation. For change projects the *change management process* is of importance. This process need to be *mature* in order to provide the proper support for a project. The maturity variable in the network is measured by assessing document maturity, metrics maturity, activity maturity, and number of assigned responsibilities.

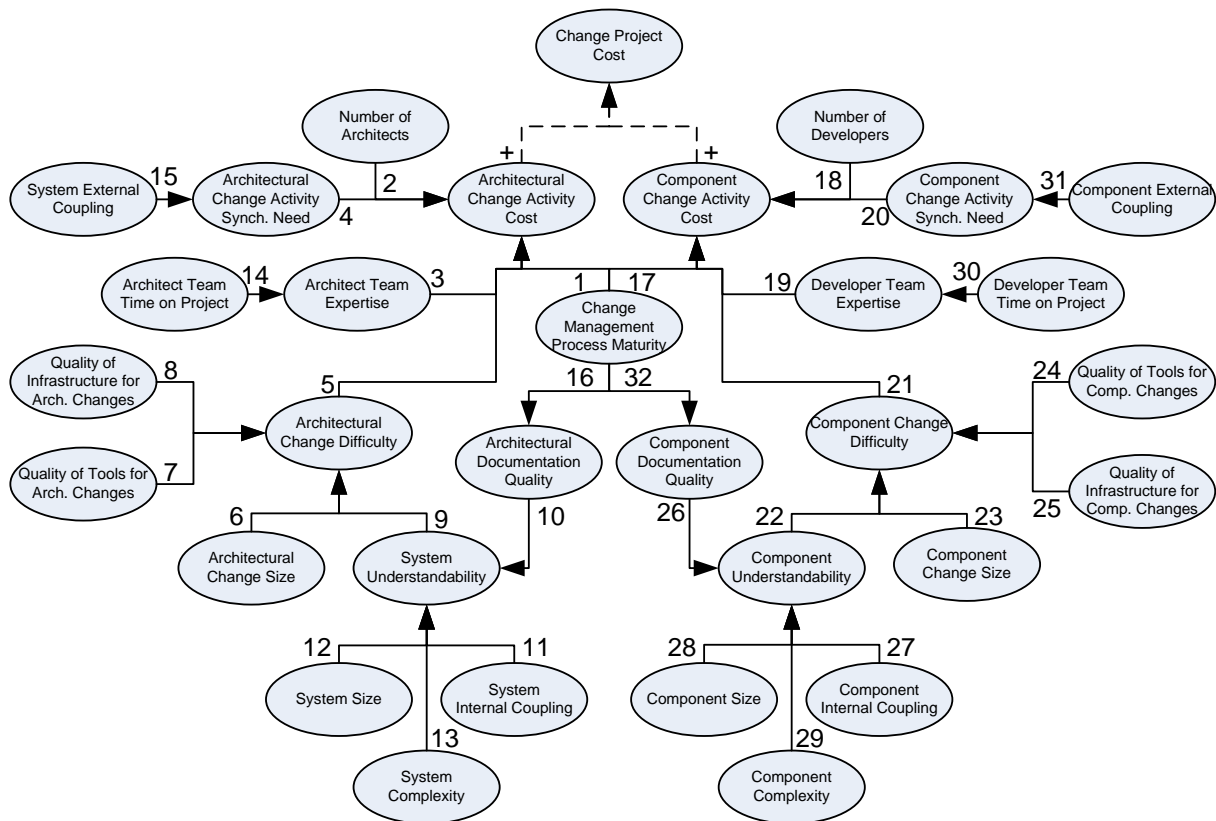


Figure 2. The Bayesian network for software change project cost analysis.

The *change organization* refers to the organizations implementing the software system modifications, i.e., the parties involved in the project, such as the customer, the vendors, consultants etc. The change organization variables in the network concerns *the number of architects and developers* involved in the project.

Architects are the people in the change project who design and modify the architecture of the enterprise systems. *Developers*, on the other hand are the ones writing and modifying the source code of the different components in the enterprise architecture. The architects and developers both have the variables *expertise* and *time on project* related to them. Expertise is measured in terms of change project experience, source code or design language experience, and system experience. Time on project refers to the amount of time a person spend in the project compared to other parallel work.

System documentation is one way for architects and developers to understand the systems, the components, and the environment. Therefore, the *architecture documentation* and the *component documentation* must

be of high *quality*, e.g. the documentation must be available, complete, accurate, consistent, and readable. Typical documents could be system rationales, requirements and design specifications, test plans, and data dictionaries.

The *system environment* and the *component environment* contain *tools*. The available tools have the intention of making the modification work easier. Although, this require the tools to be of high *quality* e.g. standardized, well known, and easy to use. The system and component environment also includes infrastructure such as platforms. *Infrastructure quality* is measured in terms of standardization level and availability.

Change projects are divided into *architectural change activities* and *component change activities*. Architectural change activities are the activities concerning modifications on an architecture level, i.e. involving several systems or components. Component change activities concern modifications to a single component. Both types of change activities have the variables *cost*, measured as number of man-hours, and *synchronization need*. The more systems, components,

people involved, and the higher the coupling between them, the higher the need of synchronization will be.

The change activities perform *technical changes to the architecture and the components*. The technical changes have two variables, *change difficulty* and *change size*. Change size for architecture is measured in number of components involved and change size for components is measured in number of lines of code involved.

Technical changes are implemented in either a *system* or a *component*. There are five variables related to these, *understandability*, *internal coupling*, *size*, *complexity*, and *external coupling*. Component external coupling concern the relations to other components, which is the same as the system internal coupling. System external coupling concerns the relations to other systems. Component size is measured in number of lines of code, and system size is an aggregate of all the components. Component internal coupling is the interdependence within a component.

5. Conditional probabilities in the network

The variables in the network, c.f. Figure 2, are all associated with a conditional probability matrix, as described in section 2. There are numerous methods available for specifying the numbers in these matrices.

One way of assigning conditional probabilities to causally linked variables is to use expert elicitation methods such as surveys, interviews, or workshops [12, 13]. However, it is both difficult and time consuming to get experts to assign probabilities directly into a network. Most people are neither familiar with Bayesian network structures nor are they used to have discussions in terms of probabilities.

Another approach is to collect empirical data for the variables and then use Bayesian network learning algorithms [9, 10, 11]. For example, measure the internal coupling, size, and understandability of a system in hundreds of cases, and then learn the probabilities of these variables based on the collected data. As stated in the related works section this approach requires a lot of time allocated from both researcher and industry, in order to collect the necessary amount of empirical data. This type of research is indeed required in the long run, but in the mean time the expert elicitation approach presented in this paper serves as a good complement.

So far, the variables in the presented network have conditional probabilities based on the knowledge from 43 experts in the field of enterprise system change management. Both the approach used for collecting the expert data and the expert elicitation method for conditional probabilities learning are presented below.

The data collection phase was divided into two parts; the first part focused on collecting data from industrial experts and the second part focused on academic experts.

The industrial experts were surveyed in two workshops with a total of 13 persons and they were given 32 statements related to the relations of the Bayesian network, see Figure 2. The academic experts were provided with two electronic surveys, one focusing on the architectural part of the network and the other on the component part. 19 people answered the architectural survey and 17 people answered the component survey.

In the workshops and the surveys the persons were also provided with questions regarding their qualifications as experts. In the workshops, two persons were excluded due to lack of expertise; both persons stated that they had not enough experience in the field. The four persons excluded from the survey either had too little experience, less than three years, they themselves said that they didn't feel certain at all about their answers, or the answer "I don't know" was given to more than eight questions.

The question asked to the experts both in the workshops and the surveys was: "How large is the effect presented in the following statements?" Then the experts were provided with statements each corresponding to a relation in the Bayesian network. See Figure 2 for all the corresponding relations.

The statements were all arranged as the following examples; "*Change management process maturity affecting architectural change activity cost*", which corresponds to the relation labeled as number 1 in the network. "*Number of architects affecting the architectural change activity cost*", corresponding to the relation labeled as number 2.

The answers provided by the experts were given on the following scale; *High effect, Low effect, No effect, I don't know*. High effect between two variables means that in **most cases** when you change the value of the affecting variable the affected variable changes as well. That is, the variables are highly correlated. Low effect between two variables means that in **some cases** when you change the value of the affecting variable the affected variable changes as well. I.e. there is low correlation between the variables. No effect between two variables means that in **no (or very few) cases** when you change the value of the affecting variable the affected variable changes as well. That is, there is no correlation between the variables. The "I don't know" answer means that either the expert did not understand the concepts in the question or the expert understood the concepts but did not have the experience to estimate the actual effect.

49 persons did provide answers to the questions either in the online surveys or in the workshops. Six of the respondents were classified as not credible, thus Table 2 only includes the answers from the 43 persons classified as credible.

Table 2. Survey results.

Statement	High effect	Low effect	No effect	I don't know	Nr of resp.
1	10	13	1	5	29
2	9	15	4	1	29
3	26	3	0	0	29
4	9	15	2	3	29
5	25	4	0	0	29
6	15	12	0	2	29
7	17	9	1	2	29
8	11	14	2	2	29
9	22	6	1	0	29
10	18	9	1	1	29
11	19	8	1	1	29
12	21	7	1	0	29
13	25	4	0	0	29
14	14	13	1	1	29
15	19	9	1	0	29
16	10	12	3	4	29
17	12	9	2	2	25
18	11	11	1	2	25
19	21	4	0	0	25
20	12	5	2	6	25
21	20	3	0	2	25
22	20	5	0	0	25
23	6	17	1	1	25
24	13	10	0	2	25
25	12	11	1	1	25
26	13	11	1	0	25
27	18	7	0	0	25
28	13	11	1	0	25
29	22	2	0	1	25
30	16	7	1	1	25
31	18	5	0	2	25
32	10	9	3	3	25

5.1 Expert elicitation method

This subsection presents the method that uses the data presented in the previous section as input for the software change cost Bayesian network. There are three so called extreme cases that help explain the outcome of the method. These cases are all related to the answer options provided.

The first extreme case would occur if the answer from all experts was *high effect* between two variables, then the CPM would look like the one presented in Table 3.

Table 3. The first extreme case, all experts state that variable X has a high effect on variable Y.

X		x_1	x_2	x_3
Y	y_1	1	0	0
	y_2	0	1	0
	y_3	0	0	1

Table 3 reads; if the value of variable X is measured to be x_1 then the probability is 1 (100 %) that the value of variable Y is y_1 . Why it is modeled with an extreme value and not e.g. with 95 % is because the uncertainty

will occur from the variety of expert answers and not from our bias. If we decide to affect the probability then we will have one more source of uncertainty which makes it difficult to derive factors influencing the correctness.

The second extreme case would occur if the answer from all experts was *no effect* between two variables, then the CPM would look like the one presented in Table 4.

Table 4. The second extreme case, all experts state that variable X has no effect on variable Y.

X		x_1	x_2	x_3
Y	y_1	0.33	0.33	0.33
	y_2	0.33	0.33	0.33
	y_3	0.33	0.33	0.33

The third extreme case would occur if the answer from all experts was *low effect* between two variables, then the CPM would look like the one presented in Table 5.

Table 5. The third extreme case, all experts state that variable X has a low effect on variable Y.

X		x_1	x_2	x_3
Y	y_1	0.666	0.167	0.167
	y_2	0.167	0.666	0.167
	y_3	0.167	0.167	0.666

At the extreme cases with either 100% or 33% probability of variable X is affecting Y the expert answers is set to have full reliability. The third extreme case is set as a midway between these other two which reduces our bias.

As can be seen in Table 2 there are no relations between two variables in the network that correspond to any of the three extreme cases. Also, some of the affected variables have more than one variable as a parent. Thus, it is necessary to combine the probabilities.

The algorithms for Bayesian network learning are based on the number of answers on each alternative for each question (excluding the "I don't know" answer). Every alternative has its own weight for modeling uncertainties in the variable relations. These weights are correlating to the relations presented in the CPMs in Table 3, Table 4, and Table 5. All answers are summed up and normalized for each individual question in consistency with the equation presented below.

$$P(y_j | x_{i,n}) = \begin{cases} \left(z_{1,n} + \frac{2}{3} z_{2,n} + \frac{1}{3} z_{3,n} \right) \frac{1}{z_{1,n} + z_{2,n} + z_{3,n}} & \text{if } i = j \\ \left(\frac{1}{6} z_{2,n} + \frac{1}{3} z_{3,n} \right) \frac{1}{z_{1,n} + z_{2,n} + z_{3,n}} & \text{if } i \neq j \\ i \in [1,2,3], j \in [1,2,3] \end{cases}$$

Here n represents the question number; i and j the ordinal choices of correlation; and z the number of answers. In this specific case the representation is as following:

- $z_{1,n}$ = number of *High* effect answers on question n
- $z_{2,n}$ = number of *Low* effect answers on question n
- $z_{3,n}$ = number of *No* effect answers on question n

Example: *Statement 15* ($n=15$). See Table 2.

How large is the effect of a system's external coupling on the need of synchronization between architectural change activities in a software change project?

Nineteen experts answered that the effect is *high*, nine experts answered that the effect is *low*, and one expert said that there is *no* effect. With the presented expert elicitation method this translates to the CPM presented in Table 6.

Table 6. The CPM for the effect between the variables in relation number 15.

$n=15$		<i>Tightly</i>	<i>Normal</i>	<i>Loose</i>
Y	<i>High</i>	0.874	0.063	0.063
	<i>Medium</i>	0.063	0.874	0.063
	<i>Low</i>	0.063	0.063	0.874

The first entry in the CPM should be read; if the system's external coupling is assessed to be tightly coupled then there is an 87 % probability that the need for synchronization in the architectural change activities is high.

When there is more than one variable affecting the outcome then there will be a joint probability relation. A representation of the joint probability between questions $1 \rightarrow m$ is represented as $P(Y/X_1, \dots, X_m)$. For each answer alternative i with relating output correlation j the joint probability is calculated as

$$P(y_j | x_{i,1}, \dots, x_{i,m}) = \sum_{n=1}^m \frac{P(y_j | x_{i,n})}{m}$$

A joint probability example between two system variables and their joint effect on a system quality is presented below.

How large effect does software system size (11) have on system understandability?

How large effect does software system internal coupling (12) have on system understandability?

For the question of how system size affects the understandability; nineteen experts answered *high*, eight answered *low* and one expert said it had *no* effect. For the internal coupling twenty-one experts answered *high*, seven answered *low* and one answered *no* effect. This translates to a CPM as presented in Table 7.

In this case the first entry should be read; if system internal coupling is assessed to be loosely coupled and system size is assessed to be small then there is an 88% probability that the system has high understandability.

Table 7. The conditional probability matrix corresponding to relation number 11 and 12 in the network after expert elicitation.

$n=12$		<i>Loose</i>			<i>Tight</i>		
$n=11$		<i>Small</i>	<i>Medium</i>	<i>Large</i>
Y	<i>High</i>	0.88	0.48	0.06
	<i>Medium</i>	0.06	0.47	0.06
	<i>Low</i>	0.06	0.05	0.88

The presented expert elicitation method does not need a large amount of empirical data like the Bayesian learning methods discussed in the related works section. Neither do the surveyed experts need to know that much about Bayesian networks or conditional probabilities to set the CPMs in the network.

6. Case Study

Once the Bayesian network for software change project cost estimation has been learned with the expert elicitation method proposed in this paper, it can be employed in decision situations.

This section illustrates the presented maintainability network's capability of predicting the costs of software change projects using two different case studies. In the first case eight projects within a large Nordic manufacturing company were studied. The second case considered two projects conducted within a Nordic consultancy firm. In this paper one project from the first case study, from now on called Project F for the sake of anonymity will be gone through more thoroughly and have all of its project specific data presented. Since the analyzed projects all are finished, data concerning their actual cost are obtainable. Thus, the network's predictions of the costs for the ten projects are listed, and compared to the actual costs of the projects specified in man hours, c.f. Table 9. Along with this, the accuracy of each one of the predictions is presented.

Project F was initiated since the company, active in the manufacturing domain, predicted increased sales and thus an increased amount of employed products. To still be able to manage the products a new function in the product management portal was needed. In

addition to this there was a need for a more secure, redundant and scalable server environment to be set up along the development of the new software application. The project had several objectives concerning improvement of the product management business, namely; to reduce the amount of hours spent on administration related to product management, to improve the support for the distributors of the products, to improve the quality of the already present services, to obtain a scalable, redundant and secure communication infrastructure and finally, to future proof the communication and server environment in terms of more long lasting functionality.

Information regarding the product management portal, its desiderated new functionality, as well as the project environment was gathered and the Bayesian network was instantiated. The information was obtained through interviews with relevant project stakeholders, e.g. the head project manager and system architect, as well as through studies of documents and archival records, e.g. the final report of the project. Furthermore a survey was adopted to back-up the already collected information and to reach project stakeholders not able to contribute to the study any other way.

In Table 8 below, the collected data for Project F is presented.

Now, having a slight broader understanding of the details concerning Project F's change environment we can in Table 9, see how the prediction of its cost turned out and compare it with the actual cost. The predictions for the remaining nine projects are also listed. Project A and B are those conducted at the consultancy firm.

As can be seen Table 9 is divided into three, from now on called, segments reading Large, Medium and Small. Further, both the actual cost of conducting the project as well as the prediction of the change cost as estimated by the Bayesian network for software change project cost analysis is presented in the table. The last column indicates the accuracy of the prediction, i.e. how close to the real value the prediction is. This can be seen as the primary measure for the quality of the predictions the proposed network is capable of. Continuing our study of project F we can see that the actual cost turned out to be 20000 man hours. The predicted cost in turn, given the empirical data from Table 8, turned out to be 17090 man hours. Thus, the accuracy of the prediction can be calculated to be 85,5 %. Studying the other projects with respect to their actual and predicted cost we can see that the accuracy measure spans from a lower bound of 57 % to 99.6 %. Moreover, 9 out of the 10 projects have an estimated cost within ranges of 16 % from the actual cost.

Table 8. Collected empirical data for software change project F.

Variables	Project F	
Change Management Process	Maturity	Not mature
Change Organizations	Number of Developers	Few
	Number of Architects	Medium
Architectural Change Activities	Cost	Medium
	Synchronization Need	N/A
Technical Changes to Architecture	Change Difficulty	Normal
	Change Size	Medium
Systems	Understandability	Normal
	Internal Coupling	Tightly
	Size	Small
	External Coupling	Tightly
	Complexity	N/A
System Change Environment	Quality of Tools	Medium
	Quality of Infrastructure	N/A
Architectural Documentation	Quality	Low
Architect Team	Expertise	High
	Time on Project	Low
Component Change Activities	Cost	Medium
	Synchronization Need	N/A
Technical Changes to Components	Change Difficulty	Normal
	Change Size	Medium
Components	Understandability	Normal
	Internal Coupling	Normal
	Size	Small
	External Coupling	Loose
	Complexity	N/A
Component Change Environment	Quality of Tools	High
	Quality of Infrastructure	N/A
Component Documentation	Quality	Medium
Developer Team	Expertise	Low
	Time on Project	High

The presented network for software change project cost analysis gives its values in the top node *Change project cost* as probabilities of the change costs being either High, Medium or Low. To enable predictions to be estimated in man hours, which is a more common and intuitive measure, the probabilities needs to be transformed into costs. Transforming probabilities into costs, with a generally implementable cost interval, is not an easy task, and a significant part of our ongoing research. In this paper the problem of probability transformation is approached by the introduction of the already mentioned segments in Table 9. These segments take into consideration whether the projects are seen as being of either Large, Medium or Small size. In order to reach the levels of accuracy presented in Table 9 the projects have to be fitted into one of the three segments before the estimation is done. This has so far only been done a posteriori to produce the wanted estimates and further research is being conducted to classify these more objectively and generally.

Table 9. Studied projects with actual cost, predicted cost and accuracy of conducted predictions.

Project size	Project nr	Actual cost	Pred. Cost	Accuracy
Large	Project A	20000	22680	0,882
	Project F	20000	17090	0,855
	Project B	14000	14460	0,968
	Project D	9100	8600	0,945
Medium	Project J	6266	5490	0,876
	Project H	6228	6310	0,987
Small	Project E	3000	2520	0,84
	Project I	2440	2250	0,922
	Project C	2300	2310	0,996
	Project G	1200	2105	0,57

Once the project has been fitted into a segment, then it is possible to make a rather accurate cost estimation. Table 10 depicts the transformation from probabilities of High, Medium, and Low cost to number of man-hours for Large, Medium, and Small projects.

Table 10. Cost intervals for categorization of a project's size.

segment \ cost	High	Medium	Low
Large	40000	6000	1000
Medium	12000	6000	1000
Small	5000	2500	1000

To be able to utilize the prediction capabilities of the Bayesian network for software change project cost analysis within the accuracy ranges from Table 9, a highly subjective prediction of the size of the project needs to be done. This high level prediction of the project size is preferably performed by project managers having experience of the project culture in the company where the prediction network is to be applied. Here a simple categorization aiming at determining if, dependent on the company, the largest projects commonly turns out to demand either 40000, 12000 or 5000 man hours. For example, if the largest projects in a company in general are considered to utilize man hours closer to the range of 40000 man hours than 12000 man hours the project should be labeled as a Large project. This means that the probabilities of a project being High, Medium or Low in the Bayesian network in Figure 2 are mapped to the man hour levels 40000, 6000 and 1000 respectively. If the project instead had been determined to be of Medium size the probabilities would have been mapped to the levels 12000, 6000 and 1000.

In a future version of the presented prediction network there will hopefully not be a need for the project managers to match each project with the segments presented in Table 10. General cost levels suitable for change cost predications independent of the project manager's subjective size segmentation of

the projects will be obtained through empirical studies where data for calibrating the change costs variable will be used.

A decision maker, typically a project leader within software change projects, will be provided with three sorts of valuable information when utilizing the presented network. Firstly, the expected costs of a set of change projects can be predicted. This means that a more rational decision making, concerning what parts of a company's project portfolio that should be prioritized, is enabled. Furthermore, it is possible for a decision maker to test different scenarios regarding the variable values in the network in order to try to lower the cost of a specific project and thereby providing increased decision support for software project portfolio prioritization. This could for example be realized by involving other developers having more suitable experience within the chosen design specific language in the project.

Secondly, when a project has been chosen from the portfolio to be initiated the network will be able to aid the project manager in the planning phase of the project. The planner can for instance get assistance to choose both architectural and development team size, as well as elaborate how the team expertise will affect the outcome.

Thirdly, the instantiated network can be used for conducting risk analysis. The network is for example able to expose parts of the project with a high risk of increasing its cost. Hence, action plans can be set up accordingly to mitigate the occurrence of these risks. This enables the resources chosen for the project to be optimized for the project specific change activities. Hence, the risk of the project exceeding its given budget and timeframes are somewhat more controllable. In Project F, an identified risk is the low quality of architectural documentation. This could for example result in that the understanding of the systems involved will cause a cost increase, and as a consequence that the project exceeds its budget.

7. Summary

This paper presented a Bayesian network customized for maintainability analysis of enterprise systems, thus the network consists of variables that can be used to predict software change project cost. IT decision makers employing the network will be able to make predictions of the change costs for different software projects. Furthermore, project managers will be aided in the planning phase of the project due to the increased amount of available information regarding the resource and competence need. The network also allows information for risk analysis in change projects

to be retrieved. In summary, support for software change project prioritization, planning, and risk analysis is enabled by the suggested network for maintainability.

8. References

- [1] N. Friedman, M. Linial, I Nachman and D. Pe'er, "Using Bayesian Networks to Analyze Expression Data", *Journal of Computational Biology*, Mary Ann Liebert, Inc., publishers, pp. 601-620, 2000.
- [2] R. Neapolitan, *Learning Bayesian Networks*, Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 2003.
- [3] F.V. Jensen, *Bayesian Networks and Decision Graphs*, Springer New York, Secaucus, NJ, USA, 2001.
- [4] B. Boehm, *Software Engineering Economics*, Prentice Hall, 1981.
- [5] P. Oman, J. Hagemester and D. Ash, "A Definition and Taxonomy for Software Maintainability", *SETL Report #91-08 TR*, University of Idaho, Moscow, ID 83843, 1992.
- [6] ISO/IEC 9126-1, *Software Engineering – Product Quality*, 2001.
- [7] N. Fenton and S. Pfleeger, *Software Metrics – A Rigorous & Practical Approach*, Second Edition, PWS Publishing Company, 1997.
- [8] L. Bass, P. Clements and R. Kazman, *Software Architecture in Practice*, the SEI Series in Software Engineering, Addison Wesley Longman, 1998.
- [9] S. Lauritzen, "The EM algorithm for graphical association models with missing data", *Computational Statistics and Data Analysis*, 19, 191-201, 1995.
- [10] P. Myllymäki, T. Silander, H. Tirri and P. Uronen, "B-Course: A Web-Based Tool for Bayesian and Causal Data Analysis", *International Journal on Artificial Intelligence Tools*, 11-3, 369-387, 2002.
- [11] A. Madsen, M. Lang, U. Kjaerulff and F. Jensen, "The Hugin Tool for Learning Bayesian Networks", *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Springer Berlin / Heidelberg, 2711, 594-605, 2004.
- [12] R. Keeney and D. Winterfeldt, "Eliciting Probabilities from Experts in Complex Technical Problems", *IEEE Transactions on Engineering Management*, Vol. 38, No. 3, 1991.
- [13] M. Druzdzel and L. van der Gaag, "Building Probabilistic Networks: Where Do the Numbers Come From?", *In IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No 4, July/August 2000.
- [14] W. Buntine, "A Guide to the Literature on Learning Probabilistic Networks from Data", *In IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No 2, April 1996.
- [15] K. Przytula and D. Thompson, "Construction of Bayesian Networks for Diagnostics", *In Aerospace Conference Proceedings*, Vol. 5, IEEE, 2000.
- [16] A. Brown, D. Carney and P. Clements, "A Case Study in Assessing the Maintainability of Large, Software-Intensive Systems", *IEEE*, 1995.
- [17] M. Broy, F. Deissenboeck and M. Pizka, "Demystifying Maintainability", *WoSQ'06*, Shanghai, China, May 21, 2006.
- [18] P. Bengtsson, N. Lassing, J. Bosch and H. van Vliet, "Analyzing Software Architectures for Modifiability", *Research Report 11/00*, 2000.
- [19] M. Matinlassi and E. Niemelä, "The Impact of Maintainability on Component-based Software Systems", *In the Proceedings of the 29th EUROMICRO Conference "New Waves in System Architecture" (EUROMICRO'03)*, 1089-6503/03, IEEE, 2003.
- [20] K. Aggarwal, Y. Singh, and J.K. Chhabra, "An Integrated Measure of Software Maintainability", *In the 2002 Proceedings Annual Reliability and Maintainability Symposium*, 0-7803-7348-0/02, IEEE, 2002.
- [21] J.C. Granja-Alvarez and M.J. Barranco-García, "A Method for Estimating Maintenance Cost in a Software Project: A case study, Software Maintenance", *Research and Practice*, Vol. 9, 161-175, John Wiley & Sons, Ltd, 1997.
- [22] T. Chan, S.L. Chung and T.H. Ho, "An Economic Model to Estimate Software Rewriting and Replacement Times", *In the IEEE Transactions on Software Engineering*, Vol. 22, No. 8, 0098-5589/96, 1996, IEEE.
- [23] P. Grubb and A. Takang, *Software Maintenance – Concepts and Practice*, Second Edition, World Scientific Publishing, 2003.
- [24] D. Smith, *Designing Maintainable Software*, Springer-Verlag, 1999.
- [25] T. Pigoski, *Practical Software Maintenance – Best Practices for Managing Your Software Investment*, Wiley Computer Publishing, John Wiley & Sons Inc., 1997.
- [26] R. Lagerström, "Analyzing System Maintainability Using Enterprise Architecture Models", *In the Journal of Enterprise Architecture*, 2007.
- [27] R. Lagerström & P. Johnson, "Using Architectural Models to Predict the Maintainability of Enterprise Systems", *In Proceedings of the 12th European Conference on Software Maintenance and Reengineering*, April 2008.