Software Process Modeling with Bayesian Belief Networks

S. Bibi, I. Stamelos

Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece
E-mails: {sbibi,stamelos}@csd.auth.gr

Abstract

Though it is widely accepted that uncertainty influences software development it is rarely captured explicitly in software models. Despite the emphasis on artifact uncertainties, process uncertainties should also be modeled. Software process modeling formalisms must be enhanced to include uncertainty values, which an environment for supporting definition and execution of process models should take under consideration. For this purpose the use of iterative Bayesian Belief Networks is suggested for representing software process models. Bayesian approach can provide a Network of software workflows and their interdependencies. Also Bayesian Networks have the mathematical background to deal with situations and problems that evolve over time and software process is one of these situations. We are not exactly sure of the steps we must follow in each situation and we need a flexible model that will deal with iterations, backward movements and incremental development. Bayesian updating of software process values is allowed by the model and is carried out during process execution. Belief values and confidence levels are continuously updated as new evidence arrives.

In this paper the structure of several BBNs based on Rational Unified Process are presented with various levels of abstraction. The models are generalized for various types of software process models from sequential to iterative, incremental models. Our target is to provide a framework in which all the necessary actions for software development are depicted. Also the sequence of these actions and their interactions will be represented. Different quantitative outputs may be obtained from such a model (such as volume of documentation, software size, defect counts, etc.). As an example, in this paper, we use a BBN that estimates software effort, based on the phases of software process. The model will be constantly updated when new information enters the model, leading gradually to more accurate predictions. Also the model has the possibility to represent the procedures of each discrete software workflow and their interactions with the procedures of the other workflows, providing an effort estimate of each separate workflow. For the application area of software effort estimation we demonstrate how the problem should be structured and how the resulting models could be further used. The implications and potential benefits of Bayesian approach are also discussed.

We concluded that Bayesian Belief Networks provide a natural, logical and probabilistic framework to depict software process modeling along with software effort estimation. BBN cover the primary objectives of models of the software process such as effective communication regarding the process. BBN are highly visual tools that can be easily explained indicating which workflows affect others. They enable evolution of the process as they can be used for sensitivity analysis in order to explore the impact of some changes in software process before actually implementing them. And the final objective which is the one more analyzed in this paper is the ability of BBN to facilitate effective planning, control and operational management of the process.

Keywords: Bayesian Belief Networks, Software Process, Software effort estimation, Bayesian Updating.

1. Introduction

The foundation of software engineering is the process layer [10]. Software process is a partially ordered set of activities, constraints and resources undertaken to manage, develop and maintain software systems. Therefore it is a critical factor for delivering quality software systems, as it aims to transform the user needs and requirements into a software product. Software process model has as a target to enable communication regarding the reuse, evolution and management of the process [3]. For this purpose a variety of software process models have been developed in order to provide a framework that will support the development of all possible types of software projects. However, successful modeling requires the identification and utilization of a suitable representation formalism which supports the modeling objectives [2] as well as uncertainty. Adapting and applying a formal descriptive model that depicts all the necessary actions of software process is as important as the process itshelf.

Narrative descriptions have been the usual form of process description. This form of description encourages instant, direct communication and guidance but lacks in documentation and organization rendering difficult the reuse and the management of the process [5]. It would be useful to have a graphical model that supports the multiple viewpoints of the model giving emphasis on effective planning, control and operational management of software process.

Various process models have been suggested in the last decades, from Waterfall model (1970) to Result Driven Incremental methodology (1997). Difficulties such as poorly defined requirements, frequent staff turnover or volatile software platforms constantly challenge software engineering projects, pointing out that sequential models are inappropriate for modelling software process. Iterative and incremental models are adapted nowadays in order to combine development activities with risk management. Such models are Spiral model, Rational Unified Process, Phased development, RDI model. When using these models it is important to have a formal notation that models the tasks that need to be performed, showing tasks that depend on other tasks and the degree of their dependence. Also it is useful to have an idea of the time and effort needed for each task, each iteration as well as for the effort needed for the completion of the project.

Bayesian Belief Networks may provide such a formal framework, complying with the above requirements. In brief, Bayesian belief networks are cause-effect graphs based on Bayesian inference, capable of modeling uncertainty. They are able to model software process in various levels of abstraction, visually present the activities performed, their dependencies and the necessary iterations of these activities. As an example, BBNs can provide as an output of each iteration, an estimation of the effort required for the completion of an item of the software process.

Studies regarding the use of BBNs in Software Engineering concern mostly Software Quality [7], [13]. In Software Cost Estimation two studies are found concerning the use of BBN [1], [12]. In [12] an empirical BBN, based on Boehm's informal classification of COCOMO cost factors was described. In [1] a semi automated way for deriving the BBN and its NPTs is suggested. These studies are based on cost estimations factors from historical data set and create estimations models without taking into consideration the software process. In this paper, we propose the use of BBNs as representation formalism of software process, and we provide an example BBN for estimating effort and updating this estimate in every phase of the development process. Also we argue that BBNs can support software process modeling by explicitly modeling the uncertainties concerning the various tasks of software process. In this model all the activities of software process are presented in various levels of abstraction. Also the dependencies among them will be presented. At each unit of time an estimation of the final effort required for the completion of the project is provided. As time evolves more information is included in the model leading to more accurate estimations. In conclusion, the suggested model will be able to represent the discrete steps of a software process, their sequence, the iterations among them and the way that each step is affected by the others. An example will be demonstrated modeling RUP. Also generalization of the suggested BBN for other process models will be discussed. In section 2, RUP is described. In section 3, a short overview of BBNs and their advantages in modeling software process are presented. In section 4, examples of their use in modeling RUP and generally software process models are described. In section 5 we conclude the paper and present ideas for future work.

2. Rational Unified Process (RUP)

RUP is a productized process developed by Rational Software, a division of IBM[]. Rational Unified Process provides a disciplined approach for assigning tasks and responsibilities during the development of a software project. It is a rigorous approach to the design and construction of software. The term “design” describes the activities related to the form and behaviour of the product as experienced by the user. The term “construction” describes the activities related to building the system and its internal mechanisms in its final form [11].

It consists of a process, tools, and best practices designed to help development organizations achieve quality, predictability, efficiency, and productivity in large and complex software projects.

RUP has been selected for the purposes of this study among the rest of software process models for a number of reasons. First and most important reason is that RUP represents the contemporary trends in the way software should be developed. Nowadays developing large software systems involves complex engineering tasks that may require iteration and rework before completion. RUP is a process structured by iterations that encourages incremental development. This process is based on the traditional waterfall model in the basic workflows but also considers volatility by re- examining each workflow if needed. Also it allows incremental development as in each of the four development phases all the necessary steps for software development are followed. As a consequence small parts of the system can be implemented before the construction phase.

Phases	Outcomes	Milestone	Evaluation criteria
Inception Phase	Vision document Project planning Risk assessment	Lifecycle Objectives	Requirements understanding Credibility of cost estimates Acceptance of the stakeholders
Elaboration Phase	Use case model Architectural prototype Revised project plan	Lifecycle Prototype	Stable → Vision document, architecture Credibility of risk elements Accurate plan for the construction phase Acceptance of the stakeholders
Construction Phase	Software product User manual Release description	Initial Operational Capability	Stable and mature product release Stakeholders ready for the transition Acceptable costs
Transition phase	Beta testing Trial use Training of users	Product release	User satisfaction Acceptable costs

Table 1: Phases of RUP process

Workflows	Actions performed	Outcomes
Business modeling	Identify concepts, entities and relationships, processes, behavior Workers and their responsibilities Glossary of terms	Business cases Business model
Requirements	List candidate requirements Understand system context Capture functional requirements Capture non- functional requirements Validate requirements	Actors Use cases Use case description
Analysis& design	Define a candidate architecture Perform architectural synthesis Refine the candidate architecture Analyze behavior Design components Design database	Analysis model (optional) Design model
Implementation	Define the organization of the code Implement classes, objects Test components Integrate subsystems	Subsystems
Test	Verify{interaction between objects, proper integration,	Tests {reliability,

	requirements implementation, defect treatment}	functionality, application and system performance}
Deployment	Package , distribute, install software	
Project management (supporting)	Frameworks{for managing software intensive projects, managing risk} Plan, staff, execute, and monitor projects.	Project plan
Configuration & change management (supporting)	Management of the updates and multiple versions Notification of the interested parties	
Environment (supporting)	Define and provide processes and tools	

Table 2: Iterative workflows of RUP process

RUP is an iterative process with four phases. Each phase involves nine workflows. The phases, their outcomes and the major milestones of each phase are described in table 1. There are nine core process workflows. Three of them are supporting workflows and the rest are core "engineering" workflows. The main goals of each workflow are presented in table 2.

3. Bayesian Belief Networks (BBN)

Bayesian Belief Networks are Directed Acyclic Graphs (DAGs), which are causal networks that consist of a set of nodes and a set of directed links between them, in a way that they do not form a cycle [8]. Each node represents a random variable that can take discrete or continuous finite, mutually exclusive values according to a probability distribution, which can be different for each node. Each link expresses probabilistic cause-effect relations among the linked variables and is depicted by an arc starting from the influencing variable (parent node) and terminating on the influenced variable (child node). The presence of links in the graph may represent the existence of direct dependency relationships between the linked variables (that some times may be interpreted as causal influence or temporal precedence). The absence of some links means the existence of certain conditional independency relationships between the variables.

The strength of the dependencies is measured by means of numerical parameters such as conditional probabilities. Formally, the relation between the two nodes is based on Baye's Rule [4]:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

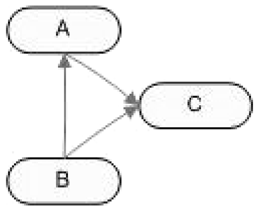


Figure 1: Simple BBN network

For each node A with parents B1, B2,…, Bn there is attached an NxM Node Probability Table (NPT), where N is the number of node states and M is the product of its cause-nodes states. In this table, each column represents a conditional probability distribution and its values sum up to 1.
Previously a general BBN has been described. Though there are BBNs that represent domains that evolve over time. In this case a discrete time stamp is introduced and a separate BBN model is considered for each unit of time. Such a local model is called a time slice. We assume the simple example coming from software engineering: For fixing a bug in a program, several actions should be taken. If the first action does not solve the problem then the next corrective action will be taken. This process repeats until the bug is fixed. If for a particular bug three corrective actions are considered then visually the BBN that models this process and estimates the probability of the bug corrected is depicted in figure. A simple time slice is depicted in figure

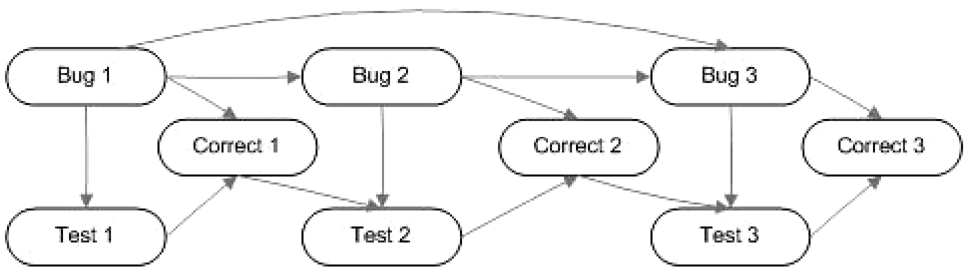
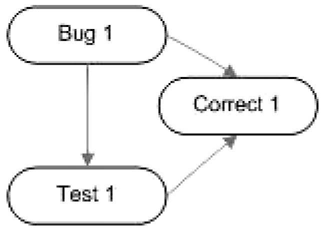


Figure 2: BBN depicting the process of correcting possible bugs



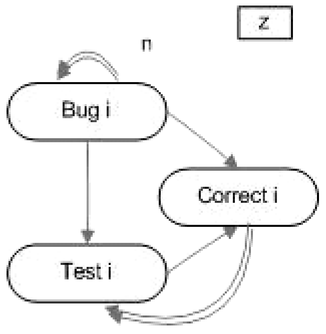


Figure 3: A case slice for bug correction **Figure 4: A compact specification of iterative models**

Regarding the advantages of BBNs in modeling software process we can mention briefly that they are suitable for
Representing the activities of software process and their dependencies for a number of reasons:

1. BBNs are able to depict the dependencies and independencies among variables and their interactions.
2. BBNs can provide the order of the actions that are necessary for software process. Each BBN shows the actions that proceed and follow each action.
3. Modern software process models are highly iterative and incremental. In both cases a descriptive model capable of representing and dealing with change and instability is necessary. BBNs have the descriptive and mathematical background to deal with procedures that evolve over time. They can represent the workflows of software process models for one iteration (waterfall model) to as many as necessary (RUP process).

Also BBNs are a suitable method for effort estimation for the following purposes:

1. BBNs offer a convenient way to solve problems that are not explained logically but rather probabilistically [9]. Software cost estimation is one of these problems: we are not sure of the factors that affect effort directly and we expect a support from statistical methods to point out the underlying relationships that appear in cost data.
2. Regarding the advantages of BBN we should mention their ability to combine expert knowledge with past historical, empirical cost data. Expert judgment becomes vital when partial or subjective information is provided about some of the important variables for example the definition of the NPTs [12].
3. Also, it is important that BBNs express uncertainty in many ways. Firstly, by providing an estimate that evolves over time. Also BBN provide estimates accompanied by probabilities considering productivity intervals, allowing flexibility in the prediction.

In general, BBNs combine visual representation with a strong mathematical background (Bayes theory, Pearl's polytree algorithm, Jensen's junction trees). They are easily interpreted, as they are represented by dependence and independence relationships, two basic human notions. Their construction is fairly easy, although we should pay some attention in the growth of the model that leads to the exponential growth of the probability matrices.

4. Methodology

In this study we utilize Bayesian Belief Networks in order to provide a representation formalism for Rational Unified Process. RUP is an iterative process, nine workflows are repeated in each of the four phases. As a consequence the whole process can be represented by a time stamped BBN model, with four time slices. Our target with the use of a time stamped BBN is to estimate the total effort needed for the completion of the project, as well as the individual effort of each phase and each workflow with the help of a BBN. Also each of the BBN presented will be generalized for other process models.

A BBN specific to the problem domain and development context must be devised. Also the specific BBN should take advantage of the unique attributes of RUP and its iterative nature. As mentioned, BBNs can be used to support experts in modeling the uncertainties in the software development process.

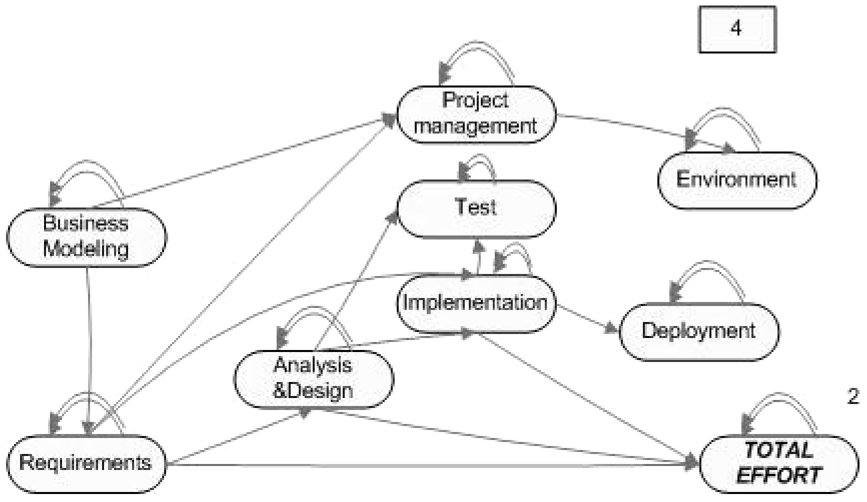


Figure 5: An initial model representing RUP process, estimating effort

The first step is to define the structure of the BBN. The significant factors that are likely to influence the effort needed for a particular project lying in the specific domain, should be identified. The cause-effect relations among these factors should also be defined. The set of factors will be the set of nodes of the BBN, the relations among them will be represented by the links whose direction will show the dependencies among them. In addition for each node an appropriate measurement scale must be defined. The outputs of the BBN can be probability distributions of interval estimates of effort. For defining the structure of the BBN that will estimate the effort needed for the completion of a software development project with the help of RUP, the significant actions that take place in each phase should be identified. Already proposed and defined factors from other cost methods will be used too.

An initial version of the BBN has as nodes some of the nine workflows. RUP can be modeled as a Kalman filter where time is counted in cases. Whenever there is evidence e entered in the model coming from the workflows, a probability P (E|e) will be provided, which is an indication from the current state of the development process estimating the expected effort for the completion of the project. While the project evolves, more information will be included in the BBN, therefore the model as the time passes will be able to give more precise effort estimates.

It would be useful to define the dependencies between the nine workflows as well as the workflows whose effort affect directly the total effort required for the completion of a project. For this purpose one or more experts need to identify the significant workflows that influence the effort required for a software project, and their interdependencies. In the model two of the supporting workflows, configuration&change management and environment, are omitted due to their limited participation at each iteration and also for simplicity purposes. It is important to keep the model as simple and interpretable as possible. Also some of the actions performed in these workflows are also embedded in other workflows (for example anticipation regarding configuration and change management is shown in every workflow). The next step is to define among the rest seven workflows which are the most critical that influence directly productivity. The requirements and the implementation workflows require significant amount of effort due to their extensive participation in the elaboration and construction phase. It is obvious that when the requirements phase is time consuming then there is a high probability that the analysis&design and the implementation phase will also require a significant amount of effort and the total effort required will be affected. The influence of requirements along with other implementation factors to the development effort has been indicated by other studies as well [1], [6].

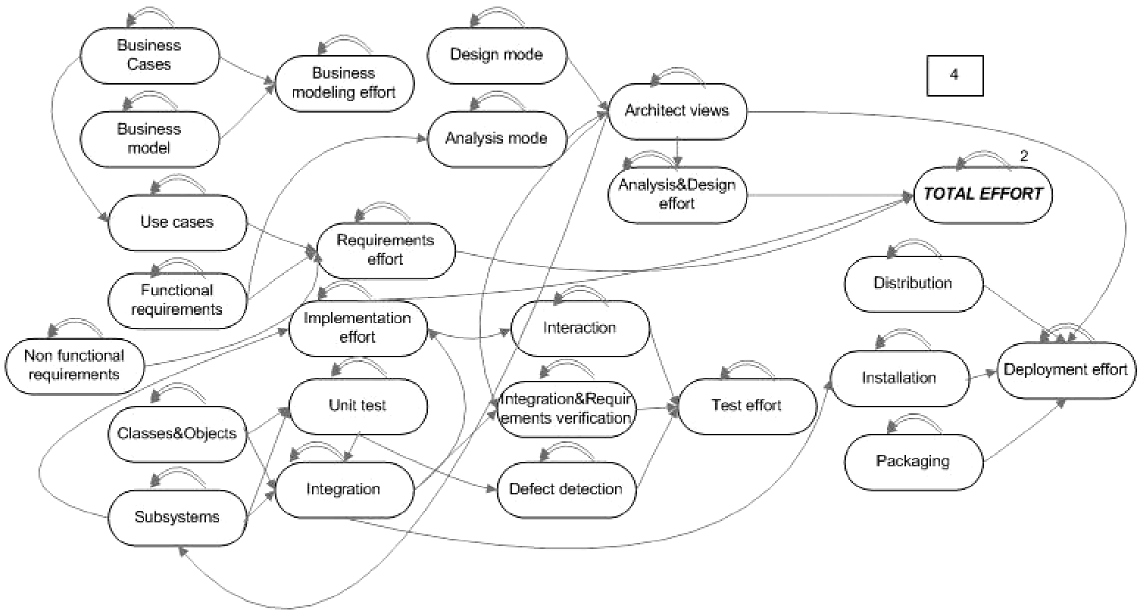


Figure 6: An analytical BBN for RUP process

The model of figure 5 could be generalized omitting the number of the iterations of the figure. If we consider one iteration the model represents Waterfall model. Also the model could represent phased development incremental or iterative where the number of the iterations of the model will depend on the number of the build and use releases.

In figure 6 a more detailed version of figure 5 is presented in order to depict the main, most important actions performed in each workflow along with their outcomes. The dependencies among the discrete actions of each workflow are also indicated by the model. It may seem a bit complicated but this BBN is representative of the way software development is performed nowadays. The development of large, complex systems involves interactions between each development phase, many iterations and constant examination of the requirements, the architecture and the implementation. This BBN can be used in order to identify the key procedures of each workflow and the way that each workflow interacts with the others. Also the sequence of each action is presented. The nodes that are parents to other nodes represent actions that occur before others actions. Also, the effort of each workflow separately is estimated. For example Analysis&design workflow effort depends on the number of architect views. The BBN of figure 6 is not suggested for total effort estimation due to its size and its complexity.

Since the final BBN for total effort estimation and its NPTs will be finally used by humans, care should be taken to keep their size as manageable as possible. A small BBN is easily explained and interpreted by humans and can be confirmed intuitively. In order to comply with the above demands, the nodes and the links directly connected with total effort were isolated. The BBN is presented in figure 7. Also the links and their direction should be carefully selected. Many links from cause nodes to an effect node increase the complexity and the clarity of the model. For this reason the nodes of requirements, analysis& design and implementation effort were inserted as additional nodes in order to avoid the direct influence of the six of the rest nodes to the total effort. A BBN with few nodes will provide larger interval estimates of effort but will be more manageable and interpretable. The model of figure 7 can be interpreted as following. When the functional, non functional requirements and the use cases numbers are known then an estimate of the requirement's effort can be made. Also when the number of architect views is known based on the analysis and design mode then the A&D effort can be estimated. These two effort estimates along with the implementation effort derived from the number of the subsystems and the expected integration time, will provide an estimate of total effort.

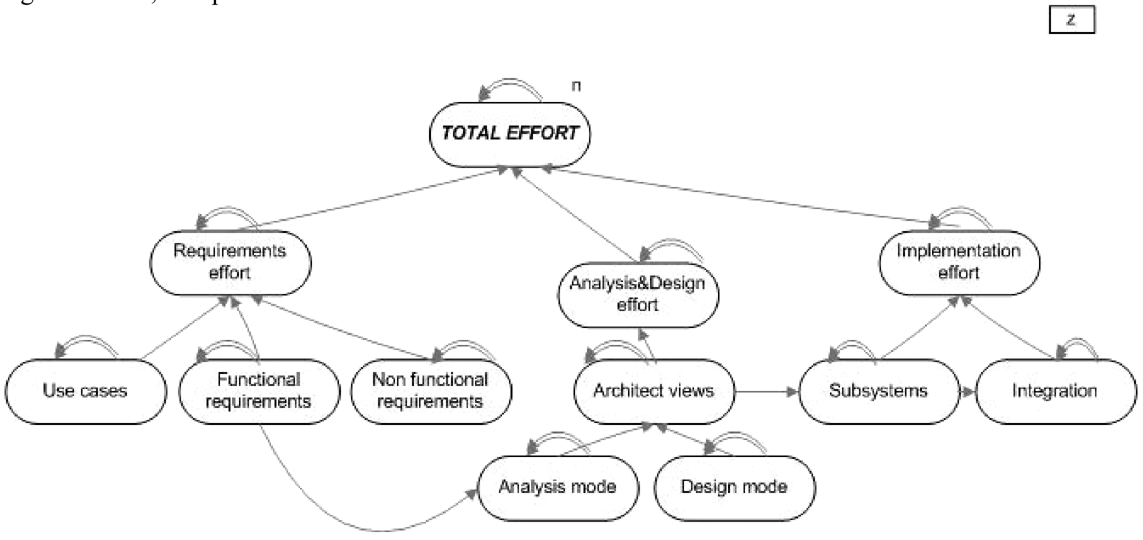


Figure 7: A generalized model that can be used for effort estimation

The model of figure 7 is general, as in every software process lifecycle model there is a requirements, an A&D and an implementation phase. In Waterfall model there is only one iteration of the model and no links from each node to itshelf (n=0, z=0). For RUP process each node is once affected by itshelf apart from total effort node that is affected by its value for two case slices. The model of RUP is repeated four times as the number of phases (n=2, z=4).

A simple empirical NPT estimating the Requirements effort based on the nodes of figure 7 is presented in table 3 in the Appendix. For example if the number of non functional requirements per use case is between 3 and 5, the number of functional requirements per use case is between 4 and 6, and the number of use cases is between 25 and 50, then there is 90% probability that the time needed to specify all kinds of requirements will be between 40 and 60 hours. These numbers are suggested from the NPT.

5. Conclusions

Software effort estimation based on process models is an important task that should be performed by all software organizations before and during the development of a project. It provides evidence of the feasibility of a project and the resources that are necessary for its completion. It is a procedure that all software improvement models include and each organization should perform.

In this paper various BBNs have been presented for modeling software process. Bayesian Belief networks are proposed for creating a model that will represent the major, time consuming activities of software process. They can be used to support expert judgement in defining the critical workflows that demand control and planning. BBNs deal with the iterative nature of most software processes exploiting additional information each time an iteration occurs, producing gradually more accurate estimates. Also according to their structure they are able to produce estimates separately for the total effort of each workflow.

The BBNs proposed previously are based on Rational Unified Process. The generalised model presented in figure 7 can be used for estimating the effort for the completion of a project created using many types of models from the Waterfall model (considering no iterations) to prototype or any other incremental, iterative model (considering the estimated number of iterations). Nevertheless for different types of projects, some differentiation of the BBNs may appear, including or excluding some nodes or links, in order to reflect different development processes. It must be stressed that it is only an intuitive model based on the experience of the experts, and the results of other studies in cost estimation.

In conclusion, Bayesian Belief Networks provide a natural, logical and probabilistic framework to combine software process modeling along with software effort estimation. BBN cover the primary objectives of models of the software process such as effective communication regarding the process. BBN are highly visual tools that can be easily explained indicating which workflows affect others. They enable evolution of the process as they can be used for sensitivity analysis in order to explore the impact of some changes in software process before actually implementing them. And the final objective which is the one more analyzed in this paper is the ability of BBN to facilitate effective planning, control and operational management of the process.

Of course BBN have to be used in real processes and estimation problems. Future work involves the creation of the models using past historical data. This process may affect somehow the structure of the suggested BBNs and will provide the Node Probability Tables. The structure of BBN indicates the suggested dependencies between the activities of each workflow and the total effort. The existence of NPTs will provide numerical parameters that show the strength of the above dependencies. With the NPTs the model will be complete and able to provide effort values apart from indicating the activities that affect directly effort.

6. References

1. S. Bibi, I. Stamelos, L. Angelis: Bayesian Belief Networks as a Software Productivity Estimation Tool. 1st Balkan Conference in Informatics, Thessaloniki, Greece, November 2003.
2. B. Curtis, M. Kellner, J. Over , " Process modeling ". Communications of the ACM 35, 9 (September 1992) 75-90.
3. P. H. Feiler, W. S. Humphrey, "Software process development and enactment: Concepts and definition". Proceedings of the Second International Conference on Software Process (February 1993) 28-40.
4. F.Jensen, Bayesian Networks and Decision Graphs, Springer, 2002.
5. M.I.Kellner, "Representation formalisms for Software Process Modeling", 4th International Software Process Workshop, Devon, UK 1989.
6. K.Maxwell, Applied Statistics for Software Managers, Prentice-Hall, PTR (2002).
7. M.Neil, N.Fenton, Predicting Software Quality using Bayesian Belief Networks,Proceedings of the 21st Annual Software Engineering Workshop ,1996,
8. J.Pearl, "Causality", Cambridge University Press, 2000.
9. J.Pearl, "Probabilistic Reasoning in Expert Systems:Networks of Plausible Inference.San Mateo, Calif.:Morgan Kaufmann.
10. R.G Pressman, "Software engineering, A Practitioner's Approach", McGraw-Hill Publishing Company, UK, 2000.
11. Rational Software Corporation, White Paper, "Rational Unified Process", Best practices for Software Development Teams, http://www.rational.com/products/whitepapers/100420.jsp
12. I.Stamelos, L.Angelis, P.Dimou, E.Sakellaris, On the use of Bayesian belief networks for the prediction of software productivity, Information and Software Technology 45 (2003) 51-60.
13. H.Ziv, D.Richardson, Constructing Bayesian-network Models of Software Testing and Maintenance Uncertainties, Proceedings of the International Conference on Software Maintenance,1997.

Appendix

Use cases number	1-10									10-25								
Number of Functional Requirements/ Use case	1-4			4-6			6-8			1-4			4-6			6-8		
Number of non Functional Requirements/ Use case	0-1	1-2	3-5	0-1	1-2	3-5	0-1	1-2	3-5	0-1	1-2	3-5	0-1	1-2	3-5	0-1	1-2	3-5
1< E ≤ 20	1	0.82	0.75	0.7	0.7	0.65	0.6	0.6	0.3	0.4	0.25	0.07	0.1	0.05	0	0.1	0.05	0
20 < E ≤ 40	0	0.18	0.30	0.3	0.3	0.25	0.3	0.3	0.6	0.5	0.6	0.73	0.6	0.7	0.8	0.6	0.4	0.3
40 ≤ E ≤ 60	0	0.0	0.05	0.0	0.0	0.10	0.1	0.1	0.1	0.1	0.15	0.17	0.3	0.25	0.2	0.3	0.65	0.7

Use cases number	25-50								
Number of Functional Requirements / Use case	1-4			4-6			6-8		
Number of non Functional Requirements / Use case	0-1	1-2	3-5	0-1	1-2	3-5	0-1	1-2	3-5
1< Effort ≤ 20	0.1	0.017	0.1	0.05	0.0	0.0	0.05	0.0	0
20 < Effort ≤ 40	0.4	0.333	0.20	0.25	0.25	0.1	0.25	0.1	0
40 ≤ Effort ≤ 60	0.6	0.65	0.7	0.7	0.85	0.9	0.7	0.9	1

Table 3: NPTs for the requirements effort node