SOFTWARE PROCESS IMPROVEMENT AND PRACTICE
Softw. Process Improve. Pract. (2009)
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/spip.418

BBN Based Approach for Improving the Software Development Process of an SME – A Case Study

Research Section

S. Bibi,[1]* I. Stamelos,[1] G. Gerolimos[2] and V. Kollias[2]
[1] *Department of Informatics, Aristotle University of Thessaloniki 54124 Thessaloniki, Greece*
[2] *Teletel S.A, 124 Kifissias Avenue, 11526 Athens, Greece*

This article proposes an approach for improving the software process of a small/medium company. The methodology is presented through a case study during which estimation models have been applied, evaluated and introduced in a telecommunication software development process. The proposed methodology uses Bayesian Belief Networks to represent the relationships among implementation, product and process metrics and their impact on the development effort. The estimation models that were derived were applied and evaluated on the on-going projects of the company. Finally, by performing the same analysis on data from the International Software Benchmarking Standards Group (ISBSG) repository, it is demonstrated how one company can utilize data from other companies when it lacks sufficient data of its own. Copyright © 2009 John Wiley & Sons, Ltd.

KEY WORDS: software process improvement; telecommunications software; quality metrics; software cost estimation

1. INTRODUCTION

Telecommunication systems (TLC) (Xu and Khoshgoftaar 2001) appear in many human activities. This results in a certain need for specialized processes and metrics for the development of custom applications. This paper focuses on a methodology for improving certain aspects of the software process in a small/medium TLC company, with emphasis on effort and quality issues.

In the literature, one can identify several software process models (Curtis *et al*. 1992; Feiler and Humphrey 1993; Chrissis *et al*. 2006) that allow for software process improvement. However, the adoption of such models by a small/medium enterprise (SME) can prove to be time consuming and often infeasible, as it requires significant experience and knowledge on how to define and implement improvement actions.

An SME can measure and analyze data of its own as a part of the improvement strategy. Effective measurement techniques are indicated as important success factors in implementing software process improvement (SPI) (Niazi *et al*. 2006). Successful SPI depends on the ability to analyze past projects and determine which parts of the process could become more efficient (Pressman 2000). Data from past projects may involve process data (e.g. effort data, team data), implementation data (e.g. development platform, language) and product data (e.g. design and code metrics). By analyzing such data certain

* Correspondence to: S. Bibi, Department of Informatics, Aristotle University of Thessaloniki 54124 Thessaloniki, Greece
† E-mail: sbibi@csd.auth.gr

useful conclusions can be drawn, such as, certain code metrics reaching values indicative of low quality, identification of conditions that may cause project delay and determination of reuse capacity of system code.

It is generally accepted and confirmed by managers and developers that for any software process it is crucial to adopt models that have both predictive and explanatory values, provided that estimates can be confirmed by intuition. Such models will exploit knowledge from past projects including important project parameters, will point out potential weaknesses of the process and eventually will complement any predictions made by software managers.

In this paper we propose a methodology for SPI in the context of small size companies and we show how it was applied in the case of a Greek TLC company that needed decision support on how to identify unsuccessful estimation practices and how to define actions in order to improve them.

Target of the study was to establish a well defined, structured and predictable software process that would lead to rapid application development for the company under study. For such purpose, issues such as accurate project effort and duration estimation, as well as potential component reuse and quality assurance are of high importance.

In our case study we adopted a graphical model (Abdel-Hamid 1989; Lai 1991) that supports the multiple viewpoints of the software process giving emphasis on effective planning, control and operational management. Bayesian Belief Networks (BBNs) provide a formal framework, complying with the above requirements. BBNs are cause-effect graphs based on Bayesian inference, capable of modeling uncertainty (Jensen 2002). They are able to model software process at various levels of abstraction and depict the activities performed and their dependencies (Bibi and Stamelos 2004; Fenton *et al*. 2007). As an example, BBNs can provide as an output the estimation of the effort, duration, size of team and code reuse required for the completion of the project (Bibi *et al*. 2003) or provide prediction of product evolution.

In this paper we apply BBNs both on the data of a TLC software company and on specific multi-organizational projects coming from the International Software Benchmarking Standards Group (ISBSG) data set (International Software Benchmarking Standards Group). SMEs (the same holds for start-ups) often have a limited amount of project data and it is reasonable to be interested in using them as much as possible and exploiting multiorganizational public data as well, i.e. data from other companies.

The company under study is a SME based in Greece and specializing in telecommunications. The study started in September 2006.

The paper is organized as follows: Section 2 discusses the related work. Section 3 presents the method of BBNs. Section 4 provides information regarding the development company and its personnel. In section 5 the case study and its phases are presented. Sections 6 and 7 discuss the results derived from single company data and from ISBSG multiorganizational data set correspondingly. In section 8 certain process enhancements are suggested. In section 9 conclusions are provided.

2. RELATED WORK

Other researchers have focused on the specialized support of the process of TLC software development mostly by suggesting software development environments (Ham *et al*. 2004) or methodologies to identify reusable components (Lee *et al*. 2003). To our knowledge there is only one study that analyzed specific development data from TLC in order to predict fault-proneness (Arisholm *et al*. 2007).

On the other hand there is a vast literature (Jørgensen 2004) in general software estimation models. Various techniques have been proposed for predicting the cost of software development with the most dominant one being expert judgment (EJ) (Jorgensen 2003). EJ, is a low cost method that depends on the experts, their level of knowledge about the methods and tools they apply. Model based techniques such as COCOMO (Boehm *et al*. 1995) model and function points (FP) (Albrecht 1979) use a formula/function based on several assumptions. Recent studies try to specialize these models based on the needs of particular software applications, e.g. DMCOMO (Marbán *et al*. 2002) for data mining applications. This fact shows the need to create estimation models parameterized for particular development fields. Regression models (RMs) (Sentas *et al*. 2005) are often among the most accurate methods. RMs provide equations

DOI: 10.1002/spip

based on several project attributes. It is easy to apply RMs in order to derive an estimate but the form of the results does not provide any information regarding the reasoning behind the estimate. Analogy based estimation is also a well known method (Chiu and Huang 2007), easily understood and applied, with the drawback that it demands the existence of a respective number of historical projects. Learning oriented techniques such as rule induction decision trees and neural networks have been employed in certain studies (Srinivisan and Fisher 1995; Mair *et al*. 2000). In these studies neural networks is among the most accurate methods, the main drawback of the method being the lack of transparency and the overfitting of the model to the data.

Bayesian Networks (BNs) have been applied in software fault prediction (Fenton *et al*. 2007) and cost estimation (Stamelos *et al*. 2003). The application of the method indicated certain advantages regarding their ability to be combined with expert judgement and provided flexible and informative estimates. The application of BNs is also found in the estimation of web applications (Mendes 2007). It is a trend to derive estimation models utilizing specialized data from a particular development field.

In this study initially we reviewed and evaluated several estimation models before selecting one of them for further application. In the company under study EJ method was used as an estimation method. The managers usually performed estimation decisions regarding resources, costs, software quality and reuse on the basis of their experience and their knowledge from previous projects. The main problem that they reported regarding their estimation procedure was the great loss of time in order to gather the appropriate evidence that would lead them to an accurate estimation. They were eager to consider some support from estimation models. One of their main considerations was that they should be able to understand the reasoning of the estimate, the form of the estimation output and should also have some evidence regarding the accuracy of the model. For these reasons after considering the advantages and disadvantages of estimation methods we selected to utilize Bayesian Networks as they can provide the following:

- A framework to model all project attributes and identify their interrelationships.
- A single model capable of multiple project attribute estimations.
- Probabilities indicating confidence of the estimation.
- Results that can be easily interpreted and confirmed by intuition.
- A formal method that can be used alone/combined/supplementary with EJ.

3. BAYESIAN BELIEF NETWORKS

BNs 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 (Jensen 2002). 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 on the basis of Bayes' Rule:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \quad (1)$$

For each node A with parents B1, B2,. . ., Bn an NxM Node Probability Table (NPT) is attached, 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.

A simple BBN estimating software effort is the one presented in Figure 1. This BBN was produced by applying the algorithm presented in (Bayesian Belief Network Software) in a part of the data collected regarding the development of certain projects of the company. In the particular example

DOI: 10.1002/spip

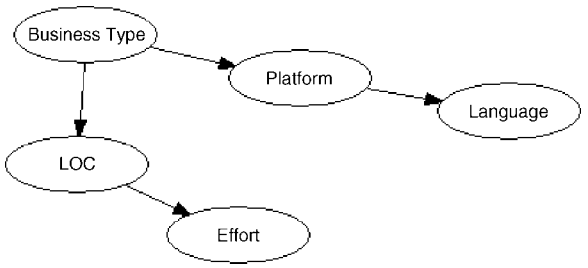

Figure 1. A BBN for software effort estimation

the variables that participate in the analysis are the business type (BT) of the application, the programming language in which the application was developed, the development platform used and the lines of code (LOC) of the application. In this example we can observe the dependencies among the variables that participate in the analysis. We can see that the BT affects directly the development platform used and the size of the application measured in LOC. In this example we also observe that the effort required for the completion of the application is directly affected by the size of the application. Therefore, the estimated effort depends on the value of LOC variable when this value is known. In case the value of LOC is not known, we can predict indirectly the value of effort from the value of BT. Knowing the value of BT we can predict the value of LOC and then the value of effort node. Attached to each node (variable) there is a node probability table (NPT) that provides probabilities for each possible value of the variable on the basis of the values of each predecessor. Attached to the node of effort there is an NPT that provides possible effort values according to the value of LOC. This NPT is presented in Table 1 and provides the following information:

- A project that is relatively small in size ($\leq$12 105 lines of code) presents 70% possibility to belong to the low effort ($\leq$5.5) interval and 30% possibility to belong to the high interval (>5.5 months). In this case the estimation points out the lower effort interval ($\leq$5.5).
- A project that is relatively large in size (>12 105 lines of code) presents 30% possibility to belong to the low effort ($\leq$5.5) interval and 70% possibility to belong to the high interval (>5.5 months). In this case the estimation points out the higher effort interval (>5.5).

Table 1. NPT example for Figure 1

LOC	$\leq$12 105	>12 105
$\leq$5.5	0.7	0.3
>5.5	0.3	0.7

It should be mentioned that apart from the effort node each of the nodes of the BBN BT, language, platform, LOC have attached an NPT table that provides estimations for the particular node on the basis of the values of its parent.

In this study the extraction of BBN is achieved with the algorithm implemented in (Bayesian Belief Network Software).

4. DEVELOPMENT COMPANY AND PERSONNEL

Teletel S.A is a small/medium Greek company that specializes in the areas of telecommunications, defense, aerospace and microelectronics. At the time of the study the company had about 35 employees. The company usually undertakes contracts to produce telecommunications and defense systems for its customers. The projects are mainly telecommunication systems, military communication applications, testing and simulation systems, radar systems, and security and interception systems. The company has long term cooperation with leading, worldwide known industries.

The company organization can be considered as advanced. Teletel is certified according to the ISO 9001 : 2000 Quality Standard. The company was not interested in obtaining CMM/I assessment and therefore not rated according to that standard.

Most members of the company are assigned to one of the following three company roles.

- Scientific personnel: A person in this role is responsible for the design, development and management of a software/hardware project.
- Technical personnel: This role is similar to that of the traditional software programmer.
- Management personnel: A person in this role is responsible for the customer contact, and usually handles contract negotiations.

All three discrete roles in the company have technical background.

DOI: 10.1002/spip

The company has a standard procedure for developing software projects based on waterfall development that contains six phases: project definition, requirements definition, design of commercial and technical solution, product development, delivery and installation, and maintenance and technical support. The procedure was forced by the international customers of the company. Teletel put effort to further standardize the development process in order to improve the quality of its products by trying to establish quality procedures that would lead to reusable aspects of code, automated requirements verification and estimation of process and product metrics. The latter activities were in their initial stage at the time of the present study.

5. THE CASE STUDY

5.1. Methodology Phases

The aim of our methodology was to design, develop, apply and evaluate models that support project management and process improvement. The methodology involves three phases, namely (a) metrics collection, b) selection, application and evaluation of formal models, and (c) specification of a new improved process.

Target of the first phase is to identify certain process, product and quality metrics that characterize the actual projects that the company develops. During the second phase, certain estimation models are applied and evaluated in order to identify the most appropriate ones. In the final phase new improved procedures are specified and intervention or control points within the existing process are proposed. At each process development phase certain actions are proposed, involving data collection or estimations.

5.2. Case Study Design

In order to apply our methodology we had to closely cooperate with three professionals, employed by the company described in Section 4. The selected experienced professionals participated also in the funded project that supported the research presented in this study and therefore were motivated to provide information. They were selected on the basis of their background and their current roles and they covered all three discrete roles mentioned: scientific personnel, management personnel and technical personnel. All participants possessed an engineering degree and a master of science in engineering. They all had experience of at least five years in the company and participated in the representative projects selected for the study.

For our study we performed eight interviews with the three participants. The overall process we followed is depicted in Figure 2.

The initial three interviews were performed during the first phase of the study. In these interviews our target was to familiarize with the development process of the company and decide where to focus estimation and process improvement efforts. For this reason we examined the standardized documents of the company in order to review the information recorded during the development of the projects. Most of the information recorded provided merely implementation details about the project under development. Managerial information could only be found spread into various documents such as the software development plan and the progress report, but no formal collection and storing of data

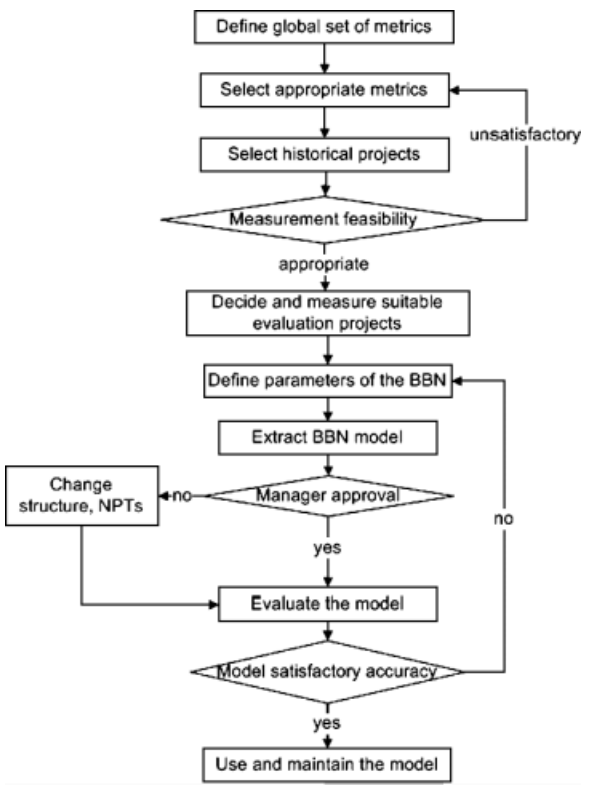

Figure 2. Methodology used for introducing the BBN model to the software process

DOI: 10.1002/spip

was performed into an estimation database that could be used in future projects. During these interviews we clarified several important issues: (a) what was the basics of the company estimation process and what changes are needed to be made, (b) what were the uses of the estimates so far in the company and what are the plans for future exploitation of the estimates, and (c) what were the estimation practices followed at the time of the research and what changes are needed to be made.

As mentioned by the project manager the estimation procedure was not performed using formal procedures or guidelines but was mostly on the basis of intuition and experience. The leading engineer for a project was also responsible for the estimation of this specific project. Estimation was expert based, but no particular formal approach was taken. However some of these experts produced estimates based on work-breakdown structure. The project manager on the basis of the type and the estimated size of the project produced a prediction that was the sum of the amounts of effort required for each development phase. Most of the times, the expert asked other experts (scientific personnel, technicians) for input about their area of expertize. On the basis of information regarding the development of projects during the period December 2003 to November 2006, we noticed that most projects were completed on time, in accordance with their schedule and the initial estimates. However, interviews revealed that a typical reason for a project being 'on target' was that the same people that participated in the estimation process were also engaged in the project and therefore responsible for any delay.

Company personnel stressed that formal decision support for estimation was needed for all three discrete roles. Current practice based on experience was time consuming and sometimes insufficient. An estimation model would help them incorporate all important project aspects in their decisions, taking into account scientific, managerial and technical information. The participants mentioned that they wanted to be able to use a model that would help them produce an estimate faster, in a way easier and safer, compared to the ad-hoc procedure they used so far.

At the time of the research the main use of the estimates was to schedule projects. The main purpose of the estimate was to commit to the customer and calculate the charges. Additionally the estimates were used for staffing the project. The variables estimated involved mainly the process, e.g. the cost, the duration and the effort required for the completion of a project. However, they also wanted help to establish estimation procedures for the quality of the developed software on the basis of code metrics. For this reason the relationship among the process metrics, the implementation metrics and the product metrics should be explored. A question that needed to be explored was whether there was a connection between implementation attributes (e.g. business type of the application, implementation language, platform) product metrics (e.g. SDL code metrics) and process metrics (e.g. effort, duration,).

In order to identify the estimating practices of the company we interviewed the project manager regarding the exact time the estimates were produced and whether any evaluation of the estimates was performed. We also asked whether the estimate was updated during the implementation of the project. The estimates in the company were prepared during an initial project proposal stage and revised only to accommodate changes in user requirements. No evaluation of the accuracy of the estimates was performed after the completion of the project. The manager mentioned that an important estimation issue was also the ability to update the estimate during the development phases of the project (e.g. requirement definition, design of commercial and technical solution, product development). Also the evaluation of the estimates both by the project manager of the particular project and by the client was in future plans.

On the basis of the responses of the participants we were able to answer the following questions:

- Which metrics can provide useful information for the company's software estimation process?
- Which projects will be considered for the generation of the estimation models and their evaluation?

Initially we defined a complete set of software process and product metrics that are proposed in the literature. After the interviews we selected the most appropriate ones, i.e. those relevant to the software that the company developed. Then we selected the historical projects that would participate in the analysis in order to define the models. In these projects we had to perform postmortem analysis. Gathering measurements from these projects proved to be a time consuming task, as for some

DOI: 10.1002/spip

completed projects information was implicit in various documents, or derived on the basis of the memory of the employees that participated in them. The employees were used as a source of data only in cases they were confident about their answers. A small variance between the data given by the employees and the actual ones did not affect our study as a result of the analysis methods that categorized the values of the data into classes. Finally, the metrics that were considered for analysis were further reduced to the ones that the company was able to collect and were derived either from the project manager of each project or from whatever information was recorded at the time of the development of the project. The next step was to select the projects that would be used for the evaluation of the estimation models. The selected on-going projects had to be similar to the historical projects used to define the estimation model. These projects had to be measured as well.

The data that were collected involved software process, product and implementation metrics. We recorded during the interviews that the basic estimation needs for the company were in the order of importance estimation of the effort needed to complete a project, estimation of the duration of the project and software reuse of components of the particular project. The assessment of the size of the team that should implement a certain project seemed an important issue as well. The metrics used to support the above mentioned estimation needs are presented in detail in section 6.

The next two interviews involved the identification of the formal model that would be used to support the estimation procedure of the company. The question explored was the following:

- Which formal models will be used to exploit these metrics and enhance current process models?

It was no surprise that the three participants showed some reluctance when asked if the proposed models would replace current estimation method (expert judgment) (Boehm and Fairley 2000). They mentioned that it was important for them to be presented with easy to understand results and also intuitively agreed with the estimation output. An evaluation of the results on on-going projects would be convincing for them provided that the estimation of the models agreed with their own estimation. Several approaches were examined in the context of this process improvement project such as regression models, association rules, decision trees and BBNs. After performing an initial analysis of the data collected, the three participants evaluated the extracted models on the basis of their estimation accuracy, the information that they provided and also the understandability of the results. Among the available formal methods for analyzing the data, Bayesian analysis was the one that seemed more suitable to the needs of the company. They pinpointed BBNs because they could handle their estimation needs, while they handled updated information and uncertainty in the estimation output. BBNs have the ability to represent dependencies among project variables, they have strong mathematical background and can detect causal relationships, unlike other estimation methods.

The final interviews focused on the deployment of the proposed models in the company software lifecycle. The main issues discussed during these interviews were the following:

- How will the estimation models be adjusted to the company's needs?
- How will the company process be updated with the models?

To answer the first question we cooperated with the managers in order to produce an appropriate BN model that would be able to fulfill their needs.

After collecting all the appropriate metrics for the historical and on-going projects the parameters of the BN model were defined. Certain important parameters that had to be defined and affected the structure and the estimation capability of the model were the following:

5.3. Causal Relationships Between Model Variables

The metrics (variables) are represented as nodes in the BBN. Each node may depend on other nodes (represented by an arc pointing towards this node) or may affect other nodes (represented by an arc from this node pointing towards other nodes). All variables should be set in an order that would affect the direction of the arcs of the BBN. Each variable could depend only on the variables before it and could affect only variables after it. For example one of the first pieces of information we are aware of when starting a software project is the business type

DOI: 10.1002/spip

of the project. On the basis of this information we can define the platform on which the project will be developed. In that case the ordering of the two variables is 'Business type', 'Platform' and therefore, the arc between the two nodes is directed towards 'Platform'.

5.4. Transformation of Continuous Variables into Categorical

This transformation is necessary because of the nature of the method (BN model) and because of the fact that it is generally safer to produce estimation in intervals (Kitchenham and Linkman 1997; Jorgensen 2003)). The decision that had to be made at this point was the number of intervals and the way the projects would be allocated to an interval. There are several formal means to decide the number of intervals. Splitting of data into intervals is usually performed by one of the following methods: equal frequency binning, equal width binning, and clustering.

5.5. The Extraction of the BBN and Intuitive Confirmation

The initial BBN is a result of statistical analysis that uses the empirical distribution of the values of each variable in order to derive the structure of the BBN and the probabilities of the NPTs. The structure of the BBN should be approved by the managers in order to be also verified and accepted intuitively. If the mangers do not approve the structure of the BBN it is possible for them to add, delete or modify certain arcs and their direction that define the dependencies among the metrics. Also in certain NPTs the probabilities for the values of each variable may be close to one another, a fact that weakens the estimation ability of the model. For this reason the refinement of the NPTs may be needed using information provided by software managers in order to change the probability values.

Eventually, the evaluation of the BBN on the on-going projects is necessary. This evaluation will show the way that the BBN should be used and whether the estimations based on it are successful and consistent with the opinion of the managers. If the results of the evaluation of the BBN are not satisfactory, certain steps must be repeated. Possible problems can be caused by the data used in the study (might require consideration of more metrics, inclusion of more projects), the definition of the parameters of the model (might require different intervals of effort, different order of variables) or inappropriate final BBN models (might require change of structure, modification of NPTs).

When the produced BBN is in agreement with the managers' needs and successfully evaluated on more recent projects, the next step is to ensure that the model is continuously updated. This can be achieved by entering new data to the model.

For each new project all necessary information should be recorded on time for two reasons: (a) it should be able to produce estimates during the software development lifecycle, and (b) it should include the knowledge derived from the project into the model when the project is completed.

6. WITHIN COMPANY DATA SET

Fifteen projects were selected to participate in the analysis. More projects were available but we preferred to exclude some projects that were very different from the average company project (e.g. had only a small software component or were quite old). Ten of these projects were already completed and were used as a training set. Five of these projects were still under development at the time of the study and were used as a test set.

In Table 2 the metrics that were collected and participated in the analysis are presented. There are three types of metrics, process metrics, product metrics and implementation metrics.

Process metrics are the effort measured in months, duration measured in months and team size measured as the number of employees who participate in a project, as well as the effort measured in man months and duration measured in months of the two development phases. The first development phase involves the initialization of the project and the analysis performed. The second development phase involves the implementation and testing procedures.

Implementation metrics involve the language in which the project is developed, the platform used and the level of reuse of historical code. Product metrics represent the values of certain code variables specialized in telecommunication software developed using the Specification and Description Language (SDL). There are three major components in SDL systems (Specification and

DOI: 10.1002/spip

Table 2. Process and product metrics

Variable	Min.	Max.	Mean	Std. Deviation	Categories
LOC (lines of code)	1235	265 000	40 734.4	80 133.0	L(≤12 105), H(>12 105)
Duration (months)	3	24	11.10	6.76	L(≤9.5), H(>9.5)
Effort (months)	3	9	5.60	2.47	L(≤5.50), H(>5.5)
P1Duration (P1 = analysis and design phase, man months)	3	19	6.67	5.32	L(≤4.5), H(>4.5)
P1Effort (man months)	3	10	5.63	2.78	L(≤5), H(>5)
P2Duration (P2 = coding and testing phase, man months)	2.5	9	5.10	2.27	L(≤5), H(>5)
P2Effort (man months)	2.65	17	5.42	4.49	L(≤3.5), H(>3.5)
Teamsize (number of people in the project)	1	4	2.20	1.03	L(≤2), H(>2)
Reuse (% of reusage of previous project products)	0	0.5	0.20	0.17	L(≤0.25), H(>0.25)
Reusability(% of the project products reused)	0	0.5	0.27	0.24	L(≤0.35), H(>0.35)
TN_B (total number of blocks)	0	23	7.00	8.65	L(≤3), H(>3)
N_P_BT (number of processes in a block type)	0	17	5.38	5.40	L(≤5), H(>5)
N_P_SYS (number of processes in a system type)	0	8	1.50	2.78	L(≤0),H(>0)
TN_P (total number of processes)	0	136	38.88	54.37	L(≤14), H(>14)
TN_ST (total number of states)	16	570	183.38	216.04	L(≤54), H(>54)
TN_PT (total number of process types)	0	9	1.75	3.01	L(≤1), H(>1)
TN_SYS (total number of systems)	0	1	0.13	0.35	L(≤0), H(>1)
TN_TMR (total number of timers)	0	36	17.38	15.32	L(≤15), H(>15)
TN_BT (total number of block types)	0	8	1.13	2.80	L(≤0), H(>0)
TN_T (total number of data types)	0	4	0.50	1.41	L(≤0), H(>0)
TN_G (total number of gates)	3	48	26.13	16.04	L(≤23), H(>23)
TN_CH (total number of channels)	0	101	18.38	37.06	L(≤0), H(>0)
TN_BIP (total number of built in procedures)	0	210	35.63	71.72	L(≤8), H(>8)
TN_Ent_VS (number of SDL entities with valid suffix)	3	939	198.63	314.87	L(≤49), H(>49)
TN_Ent_IS (number of SDL entities with invalid suffix)	3	427	135.25	183.80	L(≤38), H(>38)
AVG_N_B_PSYS (average number of blocks per system)	0	4	0.50	1.41	L(≤0), H(>0)
AVG_N_P_PBT (average number of processes per block Type)	0	17	3.13	6.27	L(≤0), H(>0)
AVG_N_ST_PPT (average number of states per process type)	0	31	8.13	11.41	L(≤4), H(>4)

Description Language (SDL) tutorial): the *process* which is an extended finite state machine that handles signals, the *block* which is a collection of concurrently running processes and represents a single processor, and the *system* which is a collection of concurrently running blocks that represent distributed, communicating computers. Different blocks and systems communicate through *channels. Timers* are used in order to ensure that certain signals are delivered at the expected time; otherwise the transmission is manipulated according to the desired manner. *Gates* are used to indicate which channels of the block types are supposed to connect.

The variables that were used in the study and the descriptive statistics for them are presented in Table 2.

As mentioned in section 3 all variables participating in the BN should be discrete. Because of the small number of projects that participated in the study we decided to map all projects into just two intervals (categories) of equal frequency of projects.

The process and product variables initially presented continuous values and therefore had to be discretized. The minimum, maximum, mean and standard deviations of the continuous values of these variables are presented in Table 2. Also in Table 2 we present the two categories considered for the discretization of each variable. The first category represents low values (L) and the second category represents high values (H). In Table 3 the implementation variables are recorded along with their possible values.

Bayesian Analysis has been applied in order to identify relationships among the variables. The

Table 3. Implementation attribute values

Variable	Values
Platform	WindowsNT&Linux, WindowsXP
Language	Other (C++,JAVA), SDL
Date	2002–2007
Business type	TLC systems, TLC applications(for medicine, defence)

DOI: 10.1002/spip

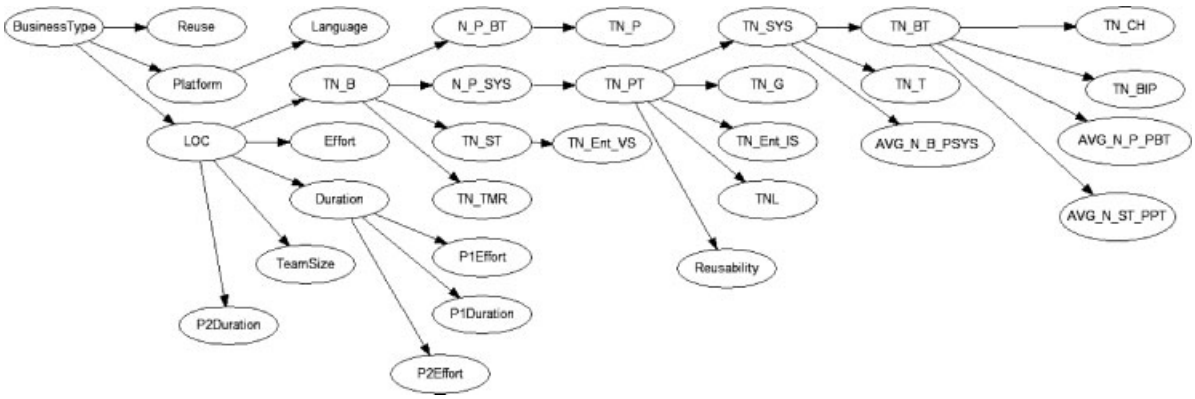

Figure 3. A BBN based on the 10 completed projects

relationships are represented by the arcs that link directly two variables and the strength of these relationships is expressed by the NPT values.

The extracted BBN is presented in Figure 3. As mentioned above, the particular tool utilized to provide the structure of the BBN uses also an order of the attributes given by the user according to which the direction of the arcs between the nodes is defined. During the initialization of a software project the first information given to a project manager is the business type of the application. It seems reasonable that the business type of the application may affect the reuse of code from past projects, because code that comes from similar business application may be considered to be the candidate for reuse. The causal relationships among the implementation attributes identified point out that the BT affects also the selection of platform and the size of the project (LOC). On the other hand, the platform affects the language in which the software is implemented. By consulting the NPTs of the BBN we found out that the level of reuse of previous software products was more than 25% in TLC business application which are mainly developed in WindowsXP and SDL language and present more than 12 105 LOC. The level of reuse of software products in other business type applications is limited to smaller percentages, and the final application is smaller in size ($\leq$12 105) compared to TLC applications.

The variable that connects implementation metrics to process and product metrics is BT. BT affects LOC that is the parent node of many process metrics and TN_B. It is reasonable that the size of a project affects the effort, duration and the size of the development team. Smaller projects require less effort and fewer people, and last fewer months compared to bigger projects. The duration of the project affects the effort and duration of the first development phase and the effort of the second phase, as suggested by managers. Projects that last 9.5 months or less require less than 5 man-months for analysis and design, and 3.5 man-months or less for coding and testing. Also the duration of the first phase of these projects is 4.5 months or less.

For this BBN the process and product metrics are affected indirectly by the BT of the application. TN_B, TN_PT, TN_SYS and TN_BT seem to affect the rest of the code metrics. TNL which is an indication of the size of a TLC project is mainly affected by TN_PT.

It should be mentioned that the BBN of Figure 3 is built on the 10 completed projects. For two of these projects product metrics were not recorded. We considered that the analysis should be repeated with only the 8 projects in order to have an objective BBN for the product metrics. The BBN that was obtained is presented in Figure 4.

This BBN is slightly different from the one of Figure 3. We can observe that in this case product metrics participate in the estimation of process metrics. The duration of a project depends on the N_P_BT and LOC. The effort required for the second development phase depends on TNL and duration. The Team Size is affected by TN_PT. The direct connection of product metrics with process metrics is very important because the estimation of certain values of product metrics during the initialization of a project is easier than the direct estimation of process metrics. In Table 4 we can see the NPT table

DOI: 10.1002/spip

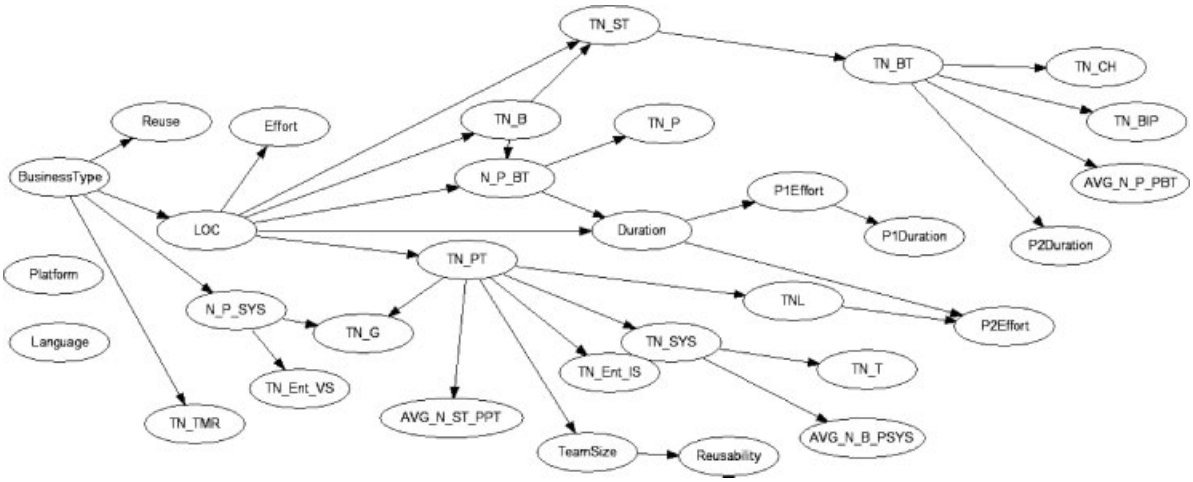

Figure 4. A BBN based on the 8 projects implemented with SDL

Table 4. NPT table for the estimation of duration (BBN, Figure 4)

N_P_BT	L		H	
LOC	L	H	L	H
L	0.69	0.42	0.42	0.44
H	0.31	0.58	0.58	0.56

for the estimation of duration.

The first column of the NPT provides the following information:

If the LOC and N_P_BT values belong to the first category (low values), there is 69% possibility that the duration will also belong to the first category (low values).

The models of Figures 3 and 4 are evaluated in terms of estimation accuracy using hit-rate which is the percentage of estimated intervals containing the actual values.

The results are presented in Table 5. The estimations coming from the node probability tables on the five projects confirm whether the estimations from the model are in agreement with the actual values. The evaluation of the model is performed on the following variables: effort, duration, team size, effort and duration of the first development phase, and effort of the second development phase. We can observe that the estimations of most of the above variables coming from the BBN of Figure 4 are in most cases (80%) in accordance with the actual values. Managers stressed also that although estimates were given in terms of only two intervals, they were still very useful for decision making, considering the increased level of project uncertainty in their development environment and the limited amount of accumulated personal experience in the company (project knowledge was possessed by only few key people in the company).

7. CROSS COMPANY DATA SET

The second data set used for the analysis comes from ISBSG, (International Software Benchmarking Standards Group). We repeated the proposed methodology on a data set that contains projects developed by different companies (cross company data). It is highly possible that a company that wants to estimate several aspects of software development will not possess a sufficient quantity of its own data. Therefore, using cross company data is a starting point for managing and estimating a software development process. Additionally cross company data can be useful when a company is adopting

Table 5. Evaluation hit-rates

Variable	Hit-rates (BBN Figure 3)(%)	Hit-rate (BBN Figure 4)(%)
effort	80	80
duration	80	100
team size	60	60
P1Effort	60	80
P1Duration	80	80
P2Effort	20	20
P2Duration	60	80

DOI: 10.1002/spip

a new technology and lacks experience. Cross company data may contain projects utilizing the particular technology and can provide support on estimation and implementation issues.

For the analysis we used ISBSG release 7. The initial data set contains 1238 projects but for our analysis we isolated the projects that had application types similar to the ones implemented by the SME company in this study. Projects with application types of 'Network management' and 'Real-time systems' were chosen for the analysis as these types were similar to the projects that the company under study developed. One project among them that had 'banking' as a business area type was excluded because it was considered to be irrelevant to the rest of the projects. Only projects that had identical measurement methods for system sizing (i.e, same method for Function Points (Albrecht 1979)) and staff sizing were considered. Also only the project attributes that did not present missing values were included. Finally the model was constructed using information from just 22 projects and 6 attributes. The model was evaluated in 5 projects of the ISBSG data set that were the most recent ones, judging from their implementation dates).

Descriptive statistics regarding the continuous variables that participated in the study are presented in Table 6. Function point and effort values were transformed to categorical values using equal width binning and four categories (low, average, high, very high).

Possible values for the categorical variables are presented in Table 7.

The BBN that is suggested in order to estimate software process development is presented in Figure 5.

Effort value is mainly affected by the amount of functionality that has to be implemented. The

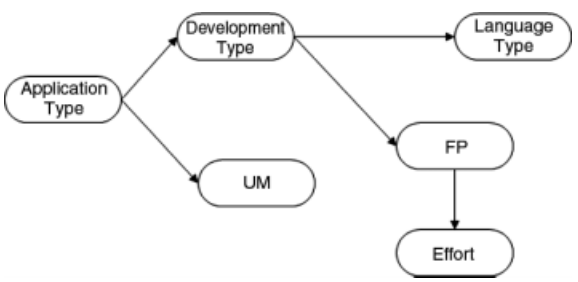

Figure 5. BBN extracted from ISBSG data set

Table 6. Continuous variables in the ISBSG data set

	Min.	Max.	Mean	Std. Deviation
Function Points	53	571	171.23	141.275
Summary work Effort	246	8713	2139.95	2226.254

Table 7. Categorical variables in ISBSG data set

Variable	Values
Development type	Enhancement, New development
Language type	3GL, 4GL
Used methodology	No, yes
Application type	Network management, real time system
Function points	L $\leq$182.5, A = 182.5–312, H = 312–441.5, VH $\geq$571
Effort	L $\leq$2362.75, A = 2362.75–4479.5, H = 4479.5–6596.25, VH $\geq$6596.25

NPT is presented in Table 8. The results of the evaluation of the model are presented in Table 9. Results are excellent for effort prediction and just encouraging for size prediction. They are satisfactory, considering the fact that training data come from multiple different companies.

The main advantage obtained from the use of ISBSG data set is that it offers a wealth of information regarding practices, tools and methodologies accompanied by process and product data. A company based on its needs can select, isolate and analyze suitable data and projects, and estimate its own new projects. In our case it was not possible to validate the BBN model of Figure 5 because the company projects were not measured in terms of FPs.

Table 8. NPT tables for effort and FP prediction

Effort/FP	1	2	3	4
1	0.67	0.52	0.25	0.24
2	0.25	0.33	0.25	0.24
3	0	0	0.25	0
4	0.08	0.15	0.25	0.524

Table 9. Evaluation hit-rates

Variable	Hit-rate(%)
Effort	100
Function points	60

DOI: 10.1002/spip

However in general, an SME would be able to assess BBN model accuracy on the basis of its own projects.

8. SOFTWARE PROCESS ENHANCEMENT

After the extraction and evaluation of the models two important questions had to be answered for the deployment of the models:

- When will the estimations be performed in future projects?
- When will the metrics be collected in future projects?

The time of the collection of the metrics was defined on the basis of the particular period during which each piece of data was generated during the development life cycle. The performance and update of the estimations were specified on the basis of the available data of each phase. Below we present certain interventions in the company's process model, at each development phase, involving data collection and estimations.

- Project definition. (collection of general project information).
  - Definition and recording of the following:
  - Development type (enhancement, re-development, new development)
  - Application type (TLC, embedded, etc.)
  - Organization Type (defense, medical, etc)
  - Requirements definition (collection of implementation metrics).
  - Definition and recording of the following:
  - Development platform
  - Programming language
  - DBMS used
  - Case tools used
  - Development methodology
  - Hardware, firmware.
  - Initial estimations of process metrics (effort, duration, team size)
- Design of commercial and technical solution (collection of size metrics)
  - Counting of application size
  - Recording of past projects reuse percentage
  - Initial estimation of product metrics (SDL metrics)
  - Adjustment of initial process estimates
- Product development (collection of code metrics)
  - LOC counting
  - Counting of comments
  - Recording of SDL metrics
  - Initial estimation of the number of faults
  - Recording of actual process metrics values
- Delivery and installation (collection of fault information)
  - Recording of actual number of faults found and short description of them
- Maintenance and technical support (gathering of fault information)
  - Recording of the number of faults during usage after the software is released

9. CONCLUSIONS

In this paper we have analyzed recent process, product and implementation data coming from a small/medium TLC software company. We suggested a methodology to model the development process using Bayesian analysis to identify relationships among the project attributes.

The proposed methodology provides the means to a small-medium software company to start collecting, analyzing and evaluating its own data, albeit the fact that only few projects have been implemented, in order to achieve a measurable, well-defined process. It can be used also to analyze cross-company data, providing alternative models to those coming from the company itself. In general, we suggest the use of BBNs in order to represent and use domain knowledge coming both from single and cross-company project data.

The results described in section 6 and feedback from the company key personnel show that the analysis performed can be a useful tool in the hands of experts. BBNs can assist in estimating under uncertainty conditions in the early stages of software development, knowing just the initial information for a project. The model can be updated any time information from new projects is available. A project manager may also continuously apply the suggested model during all phases of software development and feed new information as the project evolves in order to refine his estimations.

The suggested approach can be used for post mortem analysis in order to reach to some conclusions, for example which projects demand more effort, time, or personnel for getting completed. This is useful information for someone who is forced to make an early estimation. Such analysis can provide

DOI: 10.1002/spip

indicators of potential success or risk for on-going projects.

One crucial observation is that the analysis performed considered a limited number of projects and consequently estimation outputs were limited to just two intervals. We are in the process of gathering data for on-going projects from the same company in order to be able to stress further the overall method and be able to provide more elaborate estimates.

ACKNOWLEDGEMENT

This work has been funded by the Greek Secretary of Research and Technology, action 4.3.1, DIERGASIA project.

REFERENCES

Abdel-Hamid T. 1989. The dynamics of software project staffing: A system dynamics based simulated approach. *IEEE Transactions on Software Engineering* **15**(2): 109–112.

Albrecht A. 1979. Measuring application development productivity. Proceedings of the IBM Application Development Symposium, Monterey, California, October, 83–92.

Arisholm E, Briand L, Fuglerud M. 2007. Data mining techniques for building Fault-proneness models in telecom java software. *Forthcoming in the Proceedings of the IEEE International Symposium on Software Reliability Engineering (ISSRE)*, Sweden.

Cheng C. 2001. Bayesian Belief Network Software. http://www.cs.
ualberta.ca/~jcheng/bnpc.htm.

Bibi S, Stamelos I. 2004. Software process modelling with bayesian belief networks. *Online Proceedings of the 10th IEEE International Conference on Software METRICS*, September Chicago.

Bibi S, Stamelos I, Angelis L. 2003. Bayesian belief networks as a software productivity estimation tool. *Proceedings of the 1st Balkan Conference in Informatics*, Thessaloniki, Greece, November, 585–596.

Boehm B, Clark B, Horowitz E, Westland C, Madachy R, Selby R. 1995. In *Cost Models for Future Software Life-cycle Processes: COCOMO 2.0, Annals of Software Engineering Special Volume on Software Process and Product Measurement*, Vol. 1, Arthur JD, Henry SM (eds). J.C. Baltzer AG, Science Publishers: Amsterdam, 45–60.

Boehm B, Fairley R. 2000. Software Estimation Perspectives, IEEE Software, November/December, 22–26.

Chiu N, Huang S. 2007. The adjusted analogy-based software effort estimation based on similarity distances. *Journal of Systems and Software* **80**(4): 628–640.

Chrissis M, Konrad M, Shrum S. 2006. *CMMI: Guidelines for Process Integration and Product Improvement*, 2nd edn. NV, USA, Addison-Wesley.

Curtis B, Kellner M, Over J. 1992. Process modeling. *Communications of the ACM* **35**(9): 75–90.

Feiler P, Humphrey W. 1993. Software process development and enactment: Concepts and definition. *Proceedings of the Second International Conference on Software Process*, February 1993, Berlin, Germany, 28–40.

Fenton N, Neil M, Marsh W, Hearty P, Marquez D, Krause P, Mishra R. 2007. Predicting software defects in varying development lifecyscles using Bayesian Nets. *Information and Software Technology* **49**(1): 32–43.

Ham D, Kim J, Cho J, Ha S. 2004. MaRMI-III: A methodology for component-based development. *ETRI Journal* **26**(2): 167–180.

Web Page of the International Software Benchmarking Standards Group. www.isbsg.org.

Jensen F. 2002. *Bayesian Networks and Decision Graphs*. USA, Springer.

Jorgensen M. 2003. An effort prediction interval approach based on the empirical distribution of previous estimation accuracy. *Information and Software Technology* **45**: 123–126.

Jørgensen M. 2004. A review of studies on expert estimation of software development effort. *Journal of Systems and Software* **70**(1–2): 37–60.

Kitchenham B, Linkman S. 1997. Estimates, uncertainty and risk. *IEEE Software* **14**(3): 69–74.

Lai R. 1991. Process definition and Modelling methods. Technical report SPC-91084- N, Software productivity Consortium: Herndon, VA.

Lee W, Kwon O, Kim M, Shin G. 2003. A method and tool for identifying domain components using object usage information. *ETRI Journal* **25**(2): 121–132.

Mair C, Kadoda G, Lefley M, Phalp K, Schofield C, Shepperd M, Webster S. 2000. An investigation of machine learning based prediction systems. *Journal of Systems and Software* **53**: 23–29.

Marbán O, Amescua Seco A, Cuadrado J, García L. 2002. Cost drivers of a parametric cost estimation model for data

DOI: 10.1002/spip

mining projects (DMCOMO).*ADIS Workshop on Decision Support in Software Engineering*, Madrid, Spain.

Mendes E. 2007. The use of a bayesian network for web effort estimation.*International Conference on Web Engineering, ICWE*, Como, Italy, 90–104.

Niazi M, Wilson D, Zowghi D. 2006. Critical success factors for software process improvement implementation: an empirical study. *Software Process Improvement and Practice* **11**(2): 193–211.

Pressman R. 2000. *Software Engineering, A Practitioner's Approach*. USA, McGraw-Hill Publishing Company.

Sentas P, Angelis L, Stamelos I. 2005. Software productivity and effort prediction with ordinal regression. *Journal of Information & Software Technology* **47**(1): 17–29.

Specification and Description Language (SDL) tutorial. http://www.sdl-rt.org/.

Srinivisan K, Fisher D. 1995. Machine learning approaches to estimating software development effort. *IEEE Transactions on Software Engineering* **21**(2): 126–137.

Stamelos I, Angelis L, Dimou P, Sakellaris E. 2003. On the use of Bayesian belief networks for the prediction of software productivity. *Information and Software Technology* **45**: 51–60.

Xu Z, Khoshgoftaar T. 2001. Software quality prediction for high – Assurance network telecommunication systems. *Computer Journal* **44**(6): 557–568.

DOI: 10.1002/spip