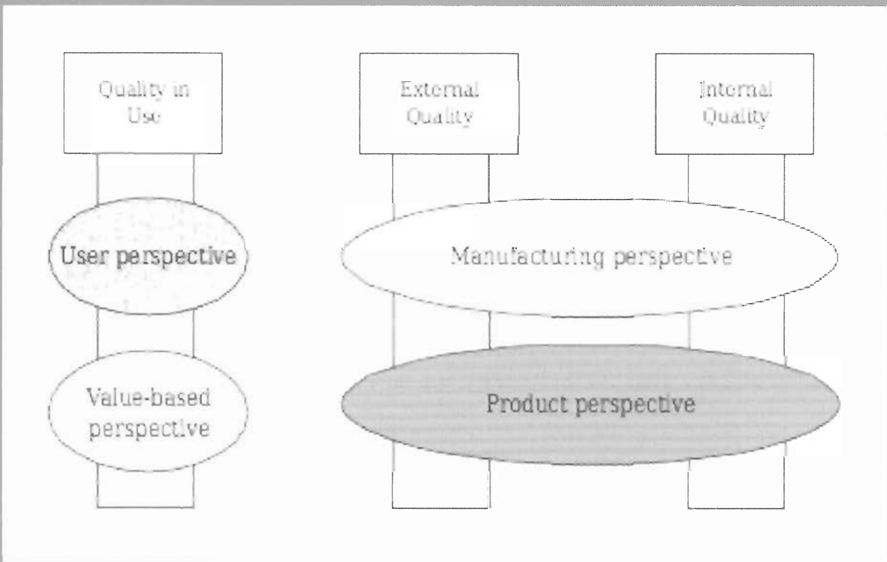
Software Quality Management XIV

Perspectives in Software Quality

Editors:
R Dawson, E Georgiadou, P Linecar,
M Ross and G Staples



The British Computer Society

Software Quality Management XIV

Perspectives
in
Software Quality

Editors:

R Dawson
Loughborough University
E Georgiadou
Middlesex University
P Linecar
London South Bank University
M Ross
Southampton Solent University & BCS Quality SG
G Staples
BCS Quality SG Chairman

THE BRITISH COMPUTER SOCIETY

BCS
THE BRITISH COMPUTER SOCIETY

ISBN 1-902505-76-X

First published 2006
Printed in Great Britain
Published by the British Computer Society

Fourteenth International Conference on Software Quality Management

SQM2006

CONFERENCE CHAIRMEN

R Dawson, E Georgiadou,
P Linecar, G Staples

CONFERENCE DIRECTOR

M Ross

INTERNATIONAL ADVISORY COMMITTEE

E Bacon	H Basson
P Bennetts	E Berki
P Burgess	D Chadwick
M Curtis	K Daily
T Daughtrey	J Dolado Cosin
A Dorling	D Edgar-Nevill
V Edgar-Nevill	I Fernandez
A Grady	T Hall
T Jackson	P Kumaran
C Long	M Love
Z Mahmood	B Myers
P Oriogun	T P Rout
K Siakas	V Sivess
J B Thompson	H Wickberg
D Wilson	

The front cover design is taken from Software Quality Model Requirements for Software Quality Engineering by Marc-Alexis Côté, Witold Suryn and Elli Georgiadou

PREFACE

This volume contains the edited proceedings of the fourteenth International Conference on Software Quality Management, SQM2006, held in Southampton, organised by the Quality Specialist Group of the British Computer Society.

The objective of this series of annual conferences is to promote international co-operation among those concerned with software quality and process improvement by creating a greater understanding of software quality issues and by sharing current research and industrial experience

The papers cover a broad spectrum of practical experience and research. The topic areas include process improvement, quality standards, metrics, estimation, product quality, methodologies, human factors, outsourcing, and quality in e-commerce systems

We would like to thank the many people who have brought this fourteenth international conference into being: the Organising Committee, the International Advisory Committee, particularly for all their hard work in reviewing both the abstracts and the final papers, and the committee members of the British Computer Society's Quality Specialist Group.

The organisers would like to thank the TickIT International, Southampton Solent University and Loughborough University for their sponsorship.

Margaret Ross and Geoff Staples
Editors

CONTENTS

Section 2: Outsourcing

Section 3: Human Factors

Building Bayesian Networks Classifiers from System Dynamics models

D. Rodríguez[1], A. Kostellos[1], M. Ruiz[2], J. Cuadrado-Gallego[3], J Dolado[4]

[1]Department of Computer Science
The University of Reading
Reading, RG6 6AY, UK
d.rodriguezgarcia@rdg.ac.uk

[2] Department of Computer Science
The University of Cádiz
C/Dr. Marañón,3 Edificio Policlínico
11002 Cádiz, Spain
mercedes.ruiz@uca.es

[3] Department of Computer Science
The University of Alcalá
Ctra. de Barcelona, km. 31,7.
28871 Alcalá de Henares, Madrid, Spain
jjcg@uah.es

[4] Department of Computer Science
The University of The Basque Country
P. Lardizabal, 1
20009 San Sebastian - Donostia, Spain
javier.dolado@ehu.es

Abstract

This paper researches on how Bayesian networks can be built from data produced from System Dynamics model simulations. Both of those techniques have already being used for decision making in software engineering processes in conjunction with domain experts. The simultaneous use of both techniques will help to overcome traditional decision making problems based strictly on project managers experience. To the extent of our knowledge, this is a new approach in software project management, since it involves the use of two orthogonal techniques.

1 Introduction

Undoubtedly effective decision making is a key point to software processes development. In recent years new decision support tools have been built, helping the managers to make more accurate predictions about the software processes [1] [2]. A high accuracy of the predictions is strongly related with a decrease in time and cost of the software processes and an increase of their quality. In this paper we combine both Bayesian Networks and System Dynamics to achieve a broader knowledge of the software process development. In our approach Bayesian networks provide a means for capturing data coming from System Dynamic simulations and therefore it makes available new knowledge.

The organisation of the paper is as follows. Section 2 provides the framework of our research, providing a review of BNs, BN classifiers and System Dynamics models. Section 3 presents and analyses our approach for building BNs and BNs classifiers from System Dynamics models. Finally, Section 4 concludes the paper and future work is outlined.

2 Related Work

2.1 Bayesian networks

Bayesian networks [3] [4] are used for modelling domains that involve uncertainty. A Bayesian network is a direct acyclic graph (DAG) which describes the structure of the network having associated a conditional probability distribution (node probability table) for each node. Each node of the model represents a domain variable that can take discrete or continuous values and the arcs among nodes represent probabilistic cause-effect relationships. The relation between two nodes X and Y is based on Bayes rule:

$$P(X|Y) = P(X|Y)\, P(X) \,/\, P(Y)$$

A Bayesian network represents a joint probability distribution over a set of random variables [5], in the sense that it can compute the conditional probabilities of some nodes given values assigned to the rest of the nodes of the network. Each node X_i has a conditional probability distribution $P(X_i \mid Parents(X_i))$ that describes the influence coming from its parents. For variables without parents, it is just marginal distribution. This allows representing the joint probability in the following way:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i| \, Parents(X_i))$$

As soon as the Bayesian network has been constructed, it can be used for performing probabilistic inference. At the beginning, the node probability tables have their own values. In case we have information-evidence for some nodes of the network, their probability tables are altered and inference is carried out over the whole network. Then the node probability tables of the remaining variables change reflecting the new information. In Figure 1, an example of a simple BN is illustrated. The BN has three nodes A, B and C. Nodes B and C are conditional dependent on node A.

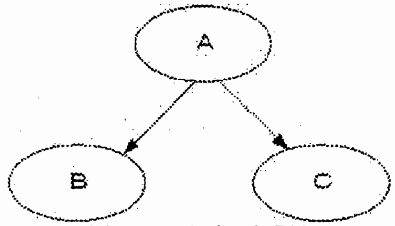


Figure 1. A simple BN

2.2 Learning Bayesian networks

Bayesian networks can be learnt from past project data. The learning procedure involves constructing the network topology and eliciting the node probability tables. Many algorithms and techniques [6] [7] have been developed that allow learning both the structure and the node probability tables from data whether having missing values or not. The typical procedure for learning BNs from historical data is shown on Figure 2.

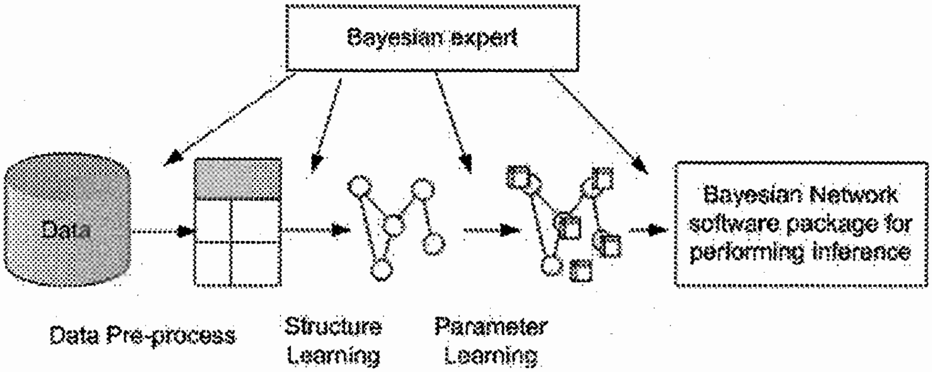


Figure 2. The typical steps needed for building BNs from data

2.3 Bayesian network classifiers

Recently, BNs have also being used for classification tasks. Their power was underestimated until Naïve Bayes classifier outperformed many state-of-the-art classifiers [7]. Naïve Bayes classifier (Figure 3) is the simplest classifier. It is represented as a BN with the class node to be the parent of all the other nodes and no arcs among attributes nodes. It learns from data the conditional probability of each attribute A_i given the class label C. Bayes theorem is applied for computing the probability of C given the $A_1,\ldots,A_n$ and then predicting the class with the highest posterior probability. For calculating the probabilities, it uses the strong assumption that every attribute A_i (leaves of the graph) is independent from the other attributes, given the value of the class variable C, i.e., root of the graph [8].

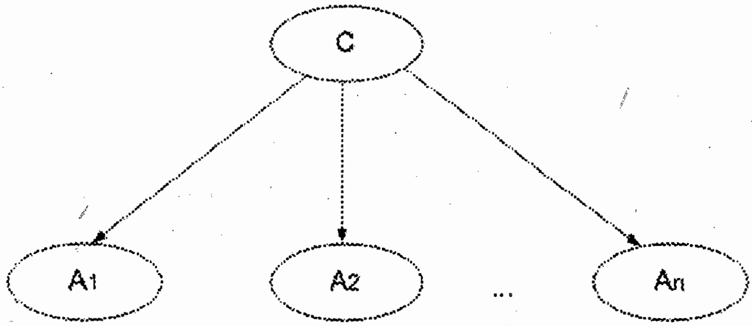


Figure 3. A simple Naïve Bayes Structure

Tree Augment Naïve Bayes (TAN) augments the Naïve Bayes structure following a tree structure. In order to increase the classification performance, there are augmented arcs between the attributes [8]. Conditionally dependencies exist among the attribute variables comparing to Naïve Bayes where attribute variables are independent. The class variable C has no parents and the attribute variables Ai have as parents the class variable and at most one other attribute. In an augment structure, an arc from A_i to A_j implies that the influence of A_i on the assessment of the class variable C also depends on the value of A_j. For example (Figure 4) the influence of the attribute A_4 on the class C depends on the value of A_3, comparing to a naïve Bayes structure where each attribute depends only on the class C.

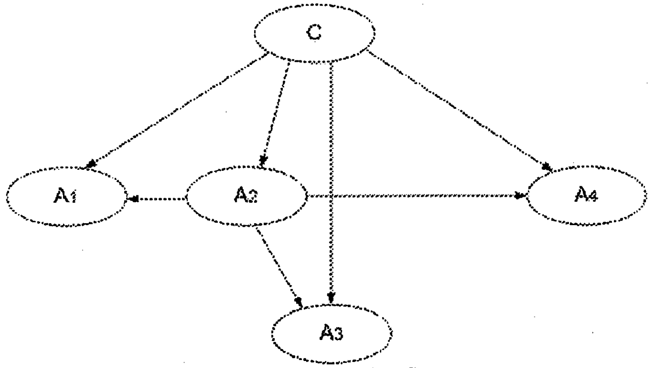


Figure 4. A simple TAN Structure

BN Augmented Naïve Bayes classifier: It was presented and evaluated by Cheng and Greiner [7]. All the attributes variables Ai of the BAN are directed connected with the classification variable C but when the class variable is removed it is a full Bayesian Network. Figure 5 shows the structure of a simple BAN classifier.

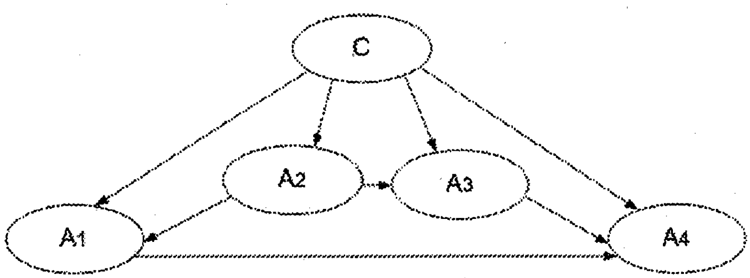


Figure 5. A simple BAN Structure

2.4 System Dynamics

The main advantage of Bayesian networks against traditional approaches (e.g. algorithmic based models) used for estimations in the area of software engineering is that they can model causality and uncertainty. According to Fenton and Neil [9] the only relevant approach to BNs is the process simulation method developed by Abdel-Hamid [10] [11] based on System Dynamics methodology [12]. System Dynamics models causality but it does not incorporate uncertainty. System Dynamics models have already being used for modelling software processes development as they provide a framework for analyzing the interactions between project activities such as code development, testing, etc. and project goals such as meeting the predefined deadlines, budget, etc.

Having knowledge of the technical factors of the software processes and the management policies would apply coupled with simulations tools facilitate organizations to improve their processes. More detailed, there are available System Dynamics software packages like VenSim, iThink, Stella, Dynamo, etc. used for building or importing already available System Dynamic models. These software tools can be used by managers to run "what if" simulations for testing different management policies without additional cost.

3. Building BNs classifiers from System Dynamics models

System Dynamics models are causal models that encompass the crucial factors influence software engineering. Although they provide simulations that valuable information is obtained, they do not incorporate the crucial notion of uncertainty facilitating better decision making for project managers. Our goal was to provide a probabilistic view to results obtained after executing the software engineering process simulations. More precisely, having knowledge of the values of some variables at time t that compose the System Dynamics model, the static BN predicts the values of the remaining variables of the model at the same time t of the simulation. Moreover, BN classifiers allow predict the class variable of the classifier given values assigned to the attributes. The final result is a decision support system that has the expert knowledge needed to take the appropriate courses of action.

In order to accomplish our goal we combine System Dynamics models and BNs. More detailed, we use a System Dynamics software package called VenSimPLE [13] for executing simulations of the public available System Dynamics model provided by [14] and we record their outcome. The dataset created from the simulations is imported to Bayesian network software package called BayesiaLab [15]. BayesiaLab is used for learning the Bayesian Network. It uses the imported dataset for creating the nodes and computing the node probability tables. Furthermore, it searches for the structure that fits the imported data best. If the structure provided by the tool does not model effectively the causal relations among the entities of the domain, the Bayesian expert (we play this role) decides about the network topology. The Bayesian expert decides which variables and causal arrows should be kept, add, or omit for best "translating" the System Dynamic models to BNs along with creating BNs having the best possible predictive performance. The typical steps for building BNs and BN classifiers from the public available System Dynamics models are:

1. Load the public available dynamic model that describes software engineering processes to VenSim software package.
2. Execute simulations of the system dynamic model using VenSim.
3. Generate the dataset: each one of the simulations produces a record with the values of the stocks, flows and auxiliary variables.
4. Pre-process data.
5. Import the dataset to BayesiaLab.
6. BayesiaLab crunches the data and learns parameters and structure of the BN classifier.
7. The Bayesian expert decides about the nodes and the relationships that will be finally used.
8. Generate and validate BNs classifiers.

3.1 The System Dynamics model

The System Dynamics model we use for executing simulations is shown in Figure 6. This dynamic model is designed to deal with errors that can be introduced during the development lifecycle of the software program. The errors can be either "programming" or "designing". The stocks of the model are: *Tasks Remaining*, *Tasks Accomplished*, and *Undiscovered Rework*. The flows of the model are: *work flow* and *rework discovery rate*. Finally, the auxiliary variables of the model are: *initial project definition*, *project is done*, *work quality* and *time to detect errors*.

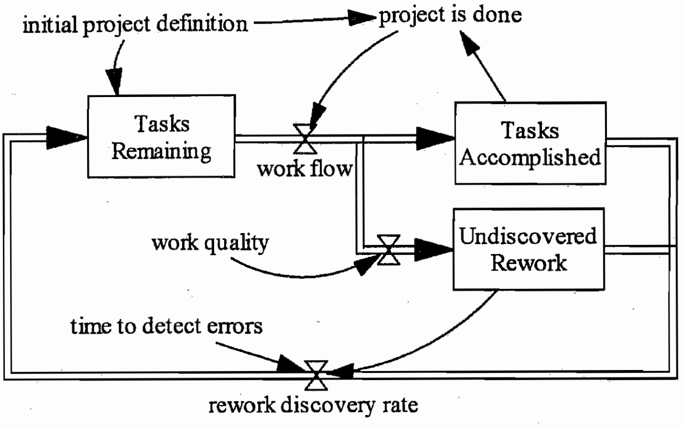


Figure 6. The public available System Dynamics model

This is a basic model that allows us to estimate the rate of work in the software house. By simulating this model, we can obtain many data for the "remaining tasks" and, therefore, we see different possible scenarios.

3.2 Building BN classifier from the System Dynamics model

We build a BN classifier (TAN) using the same dataset. Our goal is to correctly predict the value of a designated class variable given its attributes variables. To evaluate the classifiers, we split randomly the dataset produced from the simulations in two portions. The first larger portion is used only for training-learning the classifier. The second smaller portion has to be between 5% and 33% of the size of the larger portion and it is used for testing the predictive performance of the classifier.

The Dataset. In order to have the most accurate and robust predictions we spilt the dataset acquired after executing the simulations of the System Dynamics model in two portions. The first larger portion used for training the BN classifier and the second portion used for validating its performance.

Data pre-processing. The variables *Rem*, *Acc*, *URew* and *RDR* which have many values are treated as continuous variables. TAN classifier has the limitation that it can be applied only to discrete variables [16]. Therefore we have to prediscretised the continuous variables before using them for training the classifier. All continuous variables are discretised using the method of equal frequencies each of them having four intervals.

Learning process. BayesiaLab provides a set of algorithms for learning BN classifiers having different structure: Naïve architecture, Augmented Naïve architecture, Son and Spouces learning, Markov Blanket learning and Augmented Markov Blanket learning. Only the Naïve Bayes classifier and the TAN classifier connect all the nodes of the network. Moreover, research in the area of Bayesian Networks classifiers has proved that TAN classifier outperforms Naïve Bayes. Consequently, we choose to build a TAN classifier. We set the variable *Rem* as the class variable of the classifier that would predict its value. The remaining variables are the attribute variables of the classifier. BayesiaLab provides the following structure of the TAN classifier (Figure 7). The node probability tables are learnt by the BayesiaLab as well.

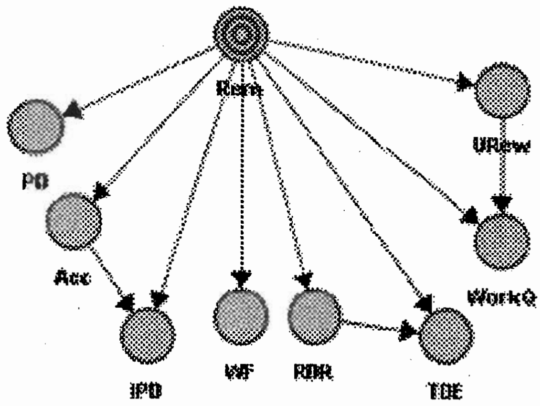


Figure 7. The BN classifier (TAN) constructed from the System Dynamics model having *Rem* as the class variable

Validation. We use the embedded tools of BayesiaLab for estimating the accuracy of the classifier (Table 1). According to the evaluation tools, the classifier (*Rem* node) achieves 76.92% total precision. The total precision is the ratio between the amount of the correct predictions and the total number of cases. Moreover, the confusion matrices are also very useful tool for interpreting the obtained results. More specifically, the confusion matrix *Accurances* shows the total number of cases versus the correct predictions. The columns show the actual values and the rows show the predicted values. Consequently, the diagonal shows all the correct predictions. According to the confusion matrix Accurances the TAN classifier predicts 15 of the 21 actual values of the interval [0-112] correctly and six values are misclassified to the interval [112-352]. Similarly it predicts correctly 17 of the 19 actual values of the interval [112-352], 15 of the 21 actual values of the interval [352-600] and 13 of the 17 actual values of the interval [600-100].

Value	*≤ 112 (21)*	*≤ 352 (19)*	*≤ 600 (21)*	*> 600 (17)*
≤ 112 (16)	15	1	0	0
≤ 352 (28)	6	17	5	0
≤ 600 (20)	0	1	15	4
> 600 (14)	0	0	1	13

Table 1. The total precision of the BN classifier (Rem node) and the Occurrences confusion matrix against the training dataset

The Reliability matrix shows the ratio between each prediction and the total number of the corresponding prediction (Table 2).

Value	*≤ 112 (21)*	*≤ 352 (19)*	*≤ 600 (21)*	*> 600 (17)*
≤ 112 (16)	93.75 %	6.25 %	0 %	0 %
≤ 352 (28)	21.42 %	60.71 %	17.8 %5	0 %
≤ 600 (20)	0 %	5 %	75 %	20 %
> 600 (14)	0 %	0 %	7.14 %	92.85 %

Table 2. Reliability confusion matrix for the BN classifier (Rem node)

Furthermore, the Precision matrix shows the ratio between each prediction and the total number of the corresponding actual values. (Table 3).

Value	*≤ 112 (21)*	*≤ 352 (19)*	*≤ 600 (21)*	*> 600 (17)*
≤ 112 (16)	71.42 %	5.26 %	0 %	0 %
≤ 352 (28)	28.57 %	%	23.8 %	0 %
≤ 600 (20)	0 %	5 %	71.42 %	23.52 %
> 600 (14)	0 %	0 %	4.76 %	76.47 %

Table3. Precision confusion matrix for the BN classifier (Rem node)

Finally we implement the simple validation method for evaluating the efficiency of the TAN classifier. A different dataset is used for testing the classifier than this used for training it. The testing dataset fulfils the requirements of the simple validation method as it is 20% of the size of the training dataset and its records are randomly selected from the original dataset. The structure of the classifier and node probability tables were learnt based on the training dataset remain fixed. In order to evaluate the performance of the classifier, we use the classifier for predicting the values of the testing dataset. The total precision of the classifier against the training dataset is 72.22%. The *Occurrences* confusion matrix (Table 4) indicates that the classifier predicts correctly: 2 of the 4 actual values of the interval [0-112], 4 of the 5 actual values of the interval [112-352], 5 of the 6 actual values of the interval [352-600] and 2 of the 3 actual values of the interval [600-100]. Tables 5 and 6 also show the Reliability and Precision matrices.

Value	*≤ 112 (4)*	*≤ 352 (5)*	*≤ 600 (6)*	*> 600 (3)*
≤ 112 (2)	2	0	0	0
≤ 352 (7)	2	4	1	0
≤ 600 (7)	0	1	5	1
> 600 (2)	0	0	0	2

Table 4. Occurrences confusion matrix

Value	*≤ 112 (4)*	*≤ 352 (5)*	*≤ 600 (6)*	*> 600 (3)*
≤ 112 (2)	100 %	0 %	0 %	0 %
≤ 352 (7)	28.57 %	57.14 %	14.28 %	0 %
≤ 600 (7)	0 %	14.28 %	71.42 %	14.28 %
> 600 (2)	0 %	0 %	0 %	100 %

Table 5. Reliability confusion matrix

Value	*≤ 112 (4)*	*≤ 352 (5)*	*≤ 600 (6)*	*> 600 (3)*
≤ 112 (2)	50 %	0 %	0 %	0 %
≤ 352 (7)	50 %	80 %	16.6 %6	0 %
≤ 600 (7)	0 %	20 %	83.33 %	33.33 %
> 600 (2)	0 %	0 %	0 %	66.66 %

Table 6. Precision confusion matrix

The performance of the TAN classifier having *Rem* variable as class variable is satisfactory. The classifier receives high precision rates against both the training and the testing dataset.

4. Conclusions and Future Work

In this paper, we have shown our approach for building Bayesian networks and Bayesian networks classifiers from System Dynamics models. Our technique provides a probabilistic view of the software process development lifecycle that it can be combined with System Dynamics models offering managers the overall view of the subject.

In our approach, we first generate a dataset using the public available System Dynamic model [13]. Then, we import the dataset to Bayesian network software package for learning the structure and the node probability tables of the BN and BN classifier. On the one hand, the BN having structure which precisely represents the cause-effect relationships of the System Dynamic model does not provide accurate predictions. The Bayesian expert has to modify its structure for increasing its predictive performance. On the other hand, the BN classifier has satisfactory predictive performance.

Summarizing, combining System Dynamics with Bayesian Networks we have a way of eliciting the knowledge that is embedded within the mental model that the software manager has about the software process. System Dynamics provides an efficient way of building mental models based on some data. The simulations allow us to have more data at our disposal and, finally, the construction of a Bayesian Network presents new relationships among variables that reflect more "expert" knowledge.

Our future work includes further research into how dynamic Bayesian networks that are an extension of Bayesian networks can be applied to software engineering. Moreover, we plan to implement more sophisticated validations techniques like simple validation and cross validation in case having limited amount of data for our BNs, in order to be sure for their predictive performance.

Acknowledgments

We would like to thank the Spanish Ministry of Science and Technology for supporting this research (Project CICYT TIN2004-06689-C03).

References

1. J. Aguilar-Ruiz, J. Riquelme, D. Rodriguez and I. Ramos (2002): 'Generation of Management Rules through Systems Dynamics and Evolutionary Computation', *Lecture Notes on Computer Science,* vol. 2559, pp. 615-628.

2. I. Stamelos, L. Angelis, P. Dimou and E. Sakellaris (2003): 'On the Use of Bayesian Belief Networks for the Prediction of Software Productivity', *Information and Software Technology*, vol. 45, pp. 51-60.

3. F. Jensen (1996): *An Introduction to Bayesian Networks*, UCL Press.

4. R.E. Neapolitan (2004): *Learning Bayesian Networks*, Prentice Hall, NJ

5. S.J. Russell and P. Norvig, *Artificial Intelligence. A Modern Approach*, Prentice-Hall, 2002

6. N. Friedman (1997): 'Learning Belief Networks in the Presence of Missing Values and Hidden Variables', *In Proceedings of the 14th International Conference on Machine Learning*, pp. 125-133.

7. J. Cheng and R. Greiner (1999): 'Comparing Bayesian Network Classifiers', *In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 101-108, Stockholm, Sweden.

8. N. Friedman, D. Geiger and M. Goldszmidt (1997): 'Bayesian Network Classifiers', *Machine Learning*, Kluwer Academic Publishers, Boston, vol 29, pp. 131-163.

9. N. Fenton and M. Neil (2000): 'Software Metrics: A Roadmap', In A. Finkelstein, editor, *The Future of Software Engineering*. ACM Press, New York, pp. 359-370.

10. T.K. Abdel-Hamid and E. Madnick (1991): *Software Project Dynamics: An*

Integrated Approach, Prentice Hall, Englewood Cliffs.

11. T.K. Abdel-Hamid (1996): 'The slippery path to productivity improvement', *IEEE Software*, vol. 13(4), pp. 43-52.

12. J. Duggan (2004): 'System Dynamics Simulation for Software Development', Department of Information Technology, National University of Ireland, Galway.

13. VenSim (2006). VenSim. Web: http://www.vensim.com/

14. M. Ruiz (2003): 'Tutorial: Software Process Improvement Achievement through Process Modeling and Simulation: An Integrative Dynamics Approach', *European Simulation and Modelling Conference ESMc's 2003*, University of Naples II, Naples, Italy.

15. BayesiaLab (2006) BayesiaLab. Web: http://www.bayesia.com/

16. N. Friedman (1998): 'The Bayesian Structural EM Algorithm', *In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 129-138.

Author's Index

This volume contains the edited proceedings of the fourteenth international conference on Software Quality Management and Process Improvement, held in Southampton, organised by the Quality Specialist Group of the British Computer Society.

The objective of this series of annual conferences is to promote international co-operation among those concerned with software quality and process improvement by creating a greater understanding of software quality issues and by sharing current research and industrial experience

The papers cover a broad spectrum of practical experience and research. The topic areas include process improvement, quality standards, metrics, estimation, product quality, methodologies, human factors, outsourcing, and quality in e-commerce systems

ISBN 1-902505-76-X