An Expert-Driven Bayesian Network Model for Simulating and Predicting Software Quality

Lukasz Radlinski
Institute of Information Technology in Management
University of Szczecin
Szczecin, Poland
lukasz@radlinski.edu.pl

Abstract— **The main goal of this work is to build an expert-driven Bayesian network model for simulating and predicting software quality. In contrast with earlier models, this model represents software quality as a hierarchy of features and their sub-features where the features are interrelated with other. It contains a range of project and process factors that influence particular quality features. It has been pre-calibrated using results from the questionnaire survey performed among software engineers and managers in various software organizations. Managers in software projects can use such model to simulate and predict various aspects of software quality, typically at the early stage of project lifecycle. Proposed may become a central part of the future decision support system aimed to analyze, understand, manage and optimize a software development process.**

***Keywords-* Bayesian network, modeling, software quality, expert knowledge, simulation, prediction**

I. INTRODUCTION

Software quality prediction is an extensively covered area of software engineering. Various models have been developed to predict different features of software quality. These models typically focus on a single aspect of software quality, for example on number of defects [4], defect proneness [2], maintainability [15] or reliability [7]. However, software quality is a combination of various features that are interrelated with each other and influenced by other factors. Unfortunately, very few predictive models, discussed in Section 2, integrate multiple aspects of software quality.

Based on the review of existing models we decided to develop a new model that would overcome their limitations. The main requirements for of the new model are the following:

- Integration of variety of quality features along with their sub-features and measures;
- Integration of project and process factors that influence quality features;
- Incorporation of expert knowledge and empirical data;
- Ability to perform various 'what-if' and 'goal-seeking' as well as advanced simulations;
- Ability to run with missing data;
- Ability to adjust the model based on new knowledge or data by the end user.

Based on an earlier analysis of different modeling techniques [13], we decided to use a Bayesian network as a formal representation of the model. With Bayesian network it is possible to satisfy all of the above requirements.

The main goal of this paper is to present selected details of the new Bayesian network for integrated software quality prediction and simulation. The model can be used in numerous analyses by answering questions such as:

- How levels of effort in various development activities influence specific quality features?
- Given a typical distribution of effort, how do environmental project factors influence software quality?
- In a project with specific project factors, how much effort should we allocate to achieve some target levels of software quality?

Earlier work on this model has been already published in [9][10][12][13]. Since the model has been evolving for about two years this paper focuses on the most recent version that satisfies all requirements stated earlier in this section. Due to limited space, this paper focuses on new results and does not cover detailed background discussion and justification that have been published in earlier papers. This paper makes the following new contributions by providing:

1. A discussion of the preferences for the expected contents of the model according to the opinions of the respondents provided during questionnaire survey.
2. The details of the most recent structure of the model defined after the questionnaire survey and based on its results.
3. The behavior of this edition of the model by discussing the results of the validation process.

This paper is organized as follows: Section 2 briefly discusses the hierarchy of software quality and revisits earlier work. Section 3 investigates the respondents preferences on the expected scope of a new model. Section 4 explains the structure of the new Bayesian network model. Section 5 discusses the behavior of this model based on the results of the validation process. Section 6 covers limitations and threats to validity of obtained results. Section 7 draws conclusions and ideas for future work.

II. Background

A. *Software Quality*

Software quality is a combination of various features. These features are often organized in a hierarchy. A variety of such hierarchies, known as software quality models, have been proposed in software engineering literature, starting from early work by Boehm and McCall at the end of 1970's. We use a hierarchy proposed in an ISO 25010 standard [6]. We chose it due to its popularity among researchers and in industry and because it has been published very recently but is based on an earlier 9126 standard – thus it can be considered as both mature and contemporary.

TABLE I. HIERARCHY OF SOFTWARE QUALITY

Features	Sub-features
Functional suitability	Functional completeness Functional correctness Functional appropriateness
Performance efficiency	Time behaviour Resource utilisation Capacity
Compatibility	Co-existence Interoperability
Usability	Appropriateness recognizability Learnability Operability User error protection User interface aesthetics Accessibility
Reliability	Maturity Availability Fault tolerance Recoverability
Security	Confidentiality Integrity Non-repudiation Accountability Authenticity
Maintainability	Modularity Reusability Analyzability Modifiability Testability
Transferability	Adaptability Installability Replaceability
Effectiveness	Effectiveness
Efficiency	Efficiency
Satisfaction	Usefulness Trust Pleasure Comfort
Freedom from risk	Economic risk mitigation Health and safety risk mitigation Environmental risk mitigation
Context coverage	Context completeness Flexibility

This hierarchy assumes three levels of quality – characteristics, sub-characteristics, and measures. Table I lists the first two groups that we call features and sub-features in our study – by changing in these names we stress that, although our predictive model is based on the ISO 25010 hierarchy of software quality, it can be relatively easy adapted to a another hierarchy, i.e., taken from a different quality model.

B. *Related Work*

Very few predictive models integrate multiple software quality features and enable comprehensive quality prediction together with the ability to perform advanced simulations. Each of these models, besides important benefits, also has some disadvantages. Wagner [16] proposed a set of models – each for a separate quality feature. Thus, this approach does not provide an integrated model with relationships between quality features. Beaver [1] proposed a model which contains a variety of links between quality features. However, this model was developed using data only from very small student projects and thus does not generalize to larger industry-scale projects. Fenton et al. [3] developed a model that incorporates empirical data and expert knowledge from industrial projects and in which quality features are linked together. However, that model contains only two quality features.

Various authors [8][17] proposed approaches or frameworks to integrated quality modeling. They do not propose a working predictive model but rather a meta-model that integrates various concepts of software quality. It can be used to support the process of building a predictive model. Such approaches may seem to be useful for developing a larger knowledge base for populating predictive models from it. However, the process of building them is time consuming. Thus, in our work we develop a predictive model directly, i.e. without an overhead of such type of framework.

III. Analysis of the Preferences on the Expected Scope of the Model

To gather data required for calibrating a new model we performed a questionnaire survey among experienced software architects and project managers. Results from the main part of that survey have been discussed in [14]. Before that main part, we asked respondents to rate five predefined versions of the model with different structures.

The main differences between model versions were related to the model complexity and the number of variables. Table II summarizes these differences. Model A was the simplest and model E was the most complex. Models B, C and D were between models A and E in terms of their complexity.

TABLE II. OVERVIEW OF MODEL VERSIONS

Characteristic \ Model	A	B	C	D	E
# process activities	3	3	1	1	3
# of process factors per activity	3	3	18	18	18
# of project factors	0	3	6	3	6
# of quality features	8	13	8	13	13
# of levels in quality hierarchy	1	2	2	3	3
reflects software composition	no	no	no	no	yes

Table III summarizes ratings for different versions of the model. We investigated six criteria: clarity, complexity, coverage, adequacy, adaptability, and usefulness. The scale available for these criteria was a range of integer numbers from '1' to '5', i.e. from low to high level of intensity of a given criterion. For all criteria, except complexity, the most desirable value was '5', i.e. that the model is clear, covers all important aspects, is adequate for a given environment, can be adapted relatively easy, and is useful. For complexity the meaning of the scale was slightly different – with a value '3' being the most desirable, and the values above '3' indicating too high level of model complexity.

We aggregated the ratings provided by respondents by calculating a weighted mean for each model and each criterion (with the necessary adjustment for the complexity). We arbitrarily defined the weights based on respondent's experience and motivation to participate in the survey. Table III shows the values of these weighted means, with the values closest to the most desirable value marked in bold.

TABLE III. RATINGS FOR MODEL VERSIONS

Model / Criterion	Weight for criterion	A	B	C	D	E
Clarity	3	**4.4**	4.0	4.1	3.5	3.0
Complexity	2	1.9	2.5	**2.8**	3.5	4.2
Coverage	1	2.3	2.9	3.3	3.8	**4.4**
Adequacy	1	3.0	2.9	**3.2**	**3.2**	2.9
Adaptability	1	3.2	2.9	**3.3**	2.9	2.3
Usefulness	5	2.6	2.4	3.1	**3.3**	2.8
SCORE	–	2.67	2.72	**3.12**	3.01	2.55

By analyzing these ratings, we can conclude that none of the model versions won in all categories. In fact, all versions except 'B', won in at least one criterion. Model A was rated as very clear but too simple for most respondents. On the other hand, model E was rated as moderately clear but too complex. The overall rating has been calculated as the weighted mean of ratings for each model. Based on this value, we can conclude that the model C was rated as the best overall, while model E as the worst overall.

IV. STRUCTURE OF THE NEW MODEL

The structure of the model presented in this section is a slightly adjusted version of model 'C' that was rated as the best overall by the respondents. Since this model is a proof-of-concept, we decided to enhance model 'C' by extending the number of process activities to three and to use all 13 quality features from the quality model proposed in the ISO 25010 standard [6]. These enhancements not only provide more functionality of the model but also gave us an opportunity to investigate the model complexity in terms of the calculation time.

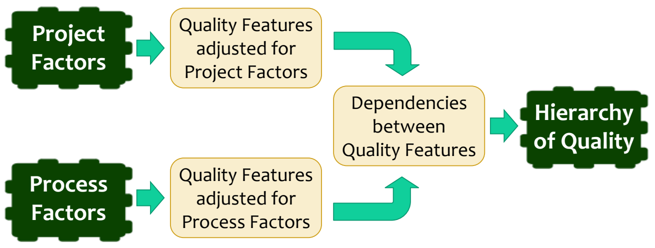
Project Factors
Quality Features adjusted for Project Factors
Dependencies between Quality Features
Hierarchy of Quality
Process Factors
Quality Features adjusted for Process Factors

Figure 1. Schematic of the proposed model

The high level schematic of this model has been illustrated in Figure 1. The core of the model consists of a set of quality features organized in a hierarchy of features, sub-features and measures, with some explicit links between main level features. These features are influenced by two groups of factors, i.e., project factors and process factors. The complete model structure and a ready-to-use model is available on-line [11].

Figure 2. illustrates the structure of the sub-network with project factors. The model contains seven project factors that describe the nature of the project. Project factors define the priors of quality features, i.e. default distributions, according to the information provided by the respondents during a questionnaire survey. A set of five 'quality in use' features (bottom of Figure 2) is much less influenced by project factors than remaining internal and external quality features. This is caused by the fact that quality in use strongly depends on the specific context/environment of use rather than on those project factors.

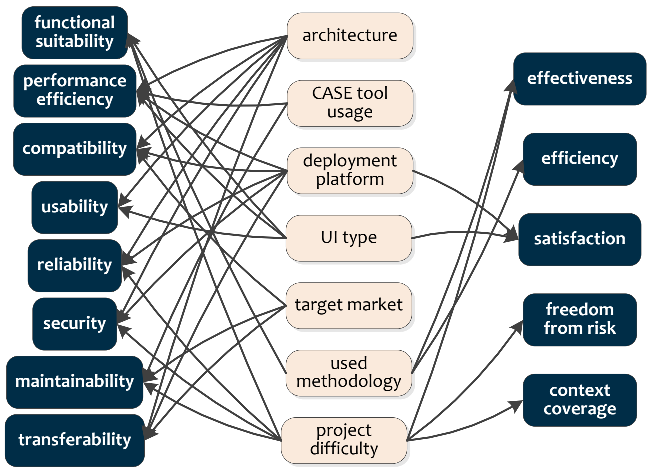

Figure 2. Structure of the sub-network with project factors and priors of quality features

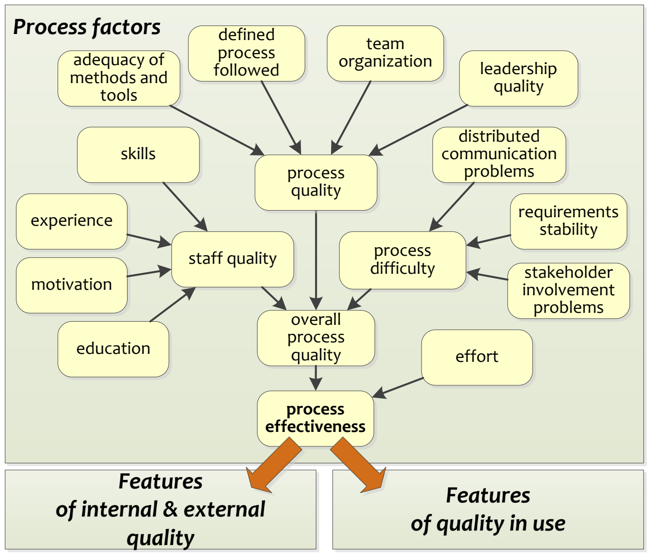

Figure 3. Structure of the sub-network with process factors

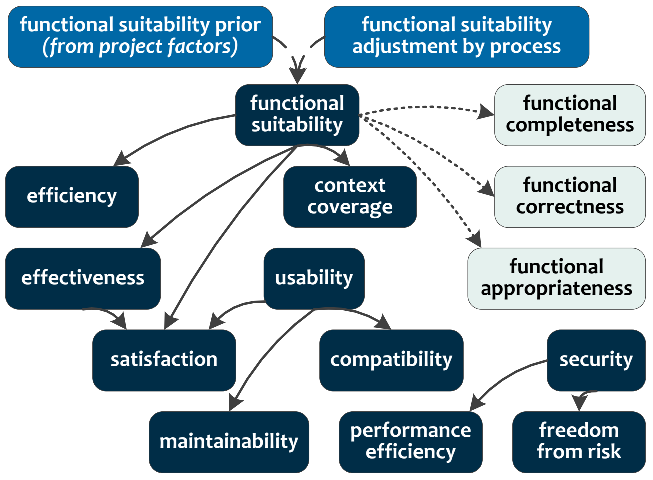

Figure 4. Links between quality features, influence of project and process factors, and a part of the quality hierarchy

Figure 3. illustrates the structure of the sub-network with the process factors. This structure has been strongly adjusted since a version used in the questionnaire survey [13]. The new structure of a converging star is clearer and more user-friendly. It enables easier adjustment directly by users – the variables that aggregate their parents are defined using expressions, most often as a 'weighted min' [5], thus adding or removing a parent variable requires only adjustment of that expression rather than manually rebuilding the whole probability table for the aggregate variable. The model contains three process sub-networks, one for each main activity, i.e. specification, development and testing.

Figure 4. illustrates the links between quality features, influence of project and process factors, and a part of the quality hierarchy. Each quality feature has its own hierarchy, i.e. a set of sub-features and measures, and is defined individually by project and process factors. Figure 4. shows all existing links between quality features but, due to a limited space, sub-features and influences from project and process factors only for an example feature – *functional suitability*. However, each quality feature is defined in a similar way with its own set of sub-features and links from sub-networks with project and process factors. Two quality features, reliability and transferability, are not directly linked with any other quality feature. It does not mean that these two features are not related with any other quality features but rather that there are no direct relationships.

V. Model Validation

To validate the developed model, we performed a variety of analyses of results provided by the model. In this paper, we discuss three of such analyses. Each of them investigates how the model behaves when an observation is entered to a single variable, i.e., what are the predictions for the other variables. Since the model is a Bayesian network, the predictions are provided not as point numeric values but as probability distributions. In our analyses, we investigated the whole probability distributions but to keep the paper concise we report the median values from predicted distributions.

All variables involved in this analysis are expressed on a 5-point ranked scale, typically from 'very low' to 'very high' but for some variables a reverse order of states is used. This ranked scale is internally transformed to a continuous scale where a state 'very low' represents a range [0, 0.2], 'low' a range [0.2, 0.4], etc. until the last state 'very high' represents a range [0.8, 1]. With such transformation it is possible to calculate statistical measures describing a probability distribution, including a median that we used in this paper.

In the first analysis, we investigated how a change of one quality feature influenced remaining quality features. First, we set an observation 'very low' to one quality feature and calculated the model. Second, we set an observation 'very high' for the same variable and calculated the model. Then, for each predicted variable we calculated the difference between median values from these two predictions (calculations) as shown in Equation 1.

$$\text{Difference}(feature\_i) = Median(feature\_i_{\text{prediction\_1}}) - Median(feature\_i_{\text{prediction\_2}}) \qquad (1)$$

TABLE IV. Predicted Relationships Between Quality Features

Quality features	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)
(1) functional suitability	–	0.06	0.20	0.20	0.12	0.14	0.17	0.11	0.55	0.53	0.38	0.21	0.46
(2) performance efficiency	0.07	–	0.11	0.07	0.09	-0.29	0.10	0.05	0.13	0.09	0.10	-0.05	0.11
(3) compatibility	0.12	0.06	–	0.36	0.11	0.10	0.22	0.11	0.16	0.14	0.18	0.12	0.15
(4) usability	0.18	0.07	0.49	–	0.10	0.10	0.44	0.09	0.19	0.18	0.28	0.16	0.18
(5) reliability	0.12	0.08	0.17	0.11	–	0.20	0.22	0.17	0.18	0.18	0.17	0.15	0.19
(6) security	0.13	-0.26	0.15	0.11	0.18	–	0.20	0.16	0.15	0.17	0.15	0.42	0.18
(7) maintainability	0.10	0.06	0.22	0.32	0.13	0.13	–	0.11	0.16	0.13	0.16	0.14	0.18
(8) transferability	0.10	0.04	0.16	0.09	0.16	0.16	0.17	–	0.15	0.15	0.15	0.10	0.15
(9) effectiveness	0.38	0.09	0.17	0.14	0.12	0.11	0.17	0.11	–	0.37	0.29	0.17	0.29
(10) efficiency	0.34	0.05	0.14	0.12	0.11	0.11	0.13	0.10	0.35	–	0.21	0.14	0.25
(11) satisfaction	0.23	0.06	0.17	0.19	0.10	0.10	0.16	0.09	0.28	0.20	–	0.12	0.21
(12) freedom from risk	0.11	-0.03	0.11	0.09	0.08	0.25	0.12	0.06	0.14	0.12	0.11	–	0.16
(13) context coverage	0.32	0.07	0.15	0.12	0.11	0.11	0.17	0.09	0.29	0.25	0.21	0.19	–

TABLE V. PREDICTIONS FOR QUALITY FEATURES DEPENDING ON OBSERVATIONS FOR PROCESS FACTORS

Process factors \ Quality features	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**	**(7)**	**(8)**	**(9)**	**(10)**	**(11)**	**(12)**	**(13)**
spec. overall process quality	0.21	0.13	0.19	0.22	-0.01	-0.02	0.09	-0.01	0.30	0.18	0.20	0.27	0.27
spec. effort	0.22	0.14	0.27	0.23	0.18	0.18	0.24	0.19	0.33	0.26	0.27	0.25	0.27
spec. process effectiveness	0.47	0.31	0.48	0.49	0.14	0.12	0.31	0.14	0.57	0.47	0.49	0.53	0.59
dev. overall process quality	-0.01	0.09	0.10	-0.01	0.21	0.17	0.30	0.20	0.17	0.09	0.12	0.06	0.17
dev. effort	0.17	0.13	0.24	0.17	0.23	0.21	0.28	0.22	0.30	0.24	0.25	0.21	0.25
dev. process effectiveness	0.12	0.23	0.34	0.13	0.47	0.40	0.59	0.45	0.47	0.32	0.37	0.26	0.43
test. overall process quality	0.17	0.00	0.19	0.16	0.19	0.22	0.07	0.20	0.07	0.20	0.16	0.08	0.07
test. effort	0.21	0.11	0.27	0.22	0.22	0.22	0.23	0.23	0.28	0.27	0.26	0.22	0.25
test. process effectiveness	0.41	0.09	0.48	0.40	0.43	0.49	0.28	0.45	0.33	0.50	0.43	0.30	0.30

TABLE VI. PREDICTIONS FOR QUALITY FEATURES DEPENDING ON OBSERVATIONS FOR PROJECT FACTORS

Project factors \ Quality features	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**	**(7)**	**(8)**	**(9)**	**(10)**	**(11)**	**(12)**	**(13)**
architecture	0.00	0.06	0.06	0.06	0.01	0.07	0.03	0.06	0.00	0.00	0.01	0.03	0.00
case tool usage	0.00	0.03	0.00	0.00	0.00	0.00	0.12	0.00	0.00	0.00	0.00	0.00	0.00
deployment platform	0.00	0.17	0.11	0.00	0.11	0.02	0.00	0.04	0.00	0.00	0.02	0.01	0.00
UI type	0.12	0.05	0.13	0.29	0.00	0.00	0.12	0.00	0.05	0.05	0.17	0.00	0.04
target market	0.00	0.00	0.03	0.00	0.00	0.00	0.11	0.04	0.00	0.00	0.00	0.00	0.00
used methodology	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.31	0.20	0.04	0.00	0.02
project difficulty	0.12	0.07	0.01	0.01	0.10	0.14	0.19	0.01	0.04	0.09	0.02	0.23	0.22

Table IV provides those differences from this first analysis. Each row represents results for quality features given a change in quality feature shown in the first column. For example, a row marked as *(1) functional suitability* contains differences in predictions for remaining quality features than occurred as a result of setting *functional suitability* to ‘very low’ and ‘very high’. A value higher than ’0.2’ can be considered as representing a positive relationship between a pair of quality features, and a value lower than ‘-0.2’ as representing a negative relationship. These results show that the model usually properly incorporates relationships identified during a questionnaire survey and discussed in [14].

However, each quality feature is at least slightly related with other quality features although often no direct relationships exist in the model. This happens because and observation in one quality feature causes revised predictions for its parents, and some of these parents then influence other quality features (i.e. there are common causes for quality features).

In the second analysis, we investigated how observations set to selected process variables influence quality features. Table V provides results for this analysis and, similarly as in the first analysis, contains the differences between the median values for quality features depending on setting observations ‘very low’ and ‘very high’ to process factors. The numbers in the first row refer to quality features according to the numbering as in Table IV. Higher values confirm that the model incorporates a relationship between particular process factor and a quality feature. These relationships are consistent with those identified and discussed in [14].

In the third analysis, we investigated how project factors influence the quality features. Table VI reports the results in the same way as in two previous analyses. However, some project factors are not expressed on a ranked scale but have labeled states. For these factors we calculated predictions by setting an observation for all possible states of a project factor (one at the time). The values of these differences also confirm that the model properly incorporates the influence of project factors on quality features as identified in the questionnaire survey [14].

VI. LIMITATIONS AND THREATS TO VALIDITY

During the work on this model and its validation, we noticed several limitations and threats to validity of obtained results. First, the relationships between quality features, illustrated in Table IV, are not always symmetrical. During the questionnaire survey, we asked respondents about such relationships without investigating the direction of the link. When building a Bayesian network, which is a directed graph, we defined directions of such link typically according to the cause-effect relationship. However, this relationship is of stochastic nature and together with other links in the model it is not possible to define links between quality features that would be symmetrical.

Second, during a questionnaire survey, respondents identified relationships between specific pairs of variables. However, very often respondents did not provide information on the details of such relationship. Even further, there were cases when one respondent provided information on the strong positive relationship between two variables, whereas according to another respondent this relationship was negative. Thus, the model cannot incorporate all information gained because of these contradictory answers.

Furthermore, in this paper, we focus on model validation that involved a change of one variable at a time. We did not report results of analyses of scenarios where multiple variables were set with observations.

Finally, model applicability is limited to software projects which follow the rationale for this model. Specifically, this includes large and long-lasting projects with a full development lifecycle. The model can be applied to other projects but after significant adjustments which may cost-ineffective.

VII. CONCLUSIONS AND FUTURE WORK

The developed Bayesian network model, discussed in this paper, is an extended and improved version of the model discussed in earlier papers. It has a simpler and clearer structure and still offers higher functionality due added useful variables. This model properly incorporates expert most knowledge gathered during the questionnaire survey as we confirmed it during the validation stage. A predefined Bayesian network model well fits the user expectations in terms of its scope, complexity and usefulness.

Although the model has been pre-calibrated, it may and should be recalibrated in the target environment. Depending on the user needs, this may involve adding or removing variables, adding or removing links, and changing the quantitative definitions of variables (e.g. the sensitivity of the changes between different variables). We believe that because of the modular structure and the usage of expressions [5] such adjustments are fairly simple.

In the future, we plan to extend the model by using detailed software measures, i.e., metrics. During the questionnaire survey we were aware that it would be difficult to obtain real data on them. We hope that, after presenting the results from the proof-of-concept model, the companies would be willing to cooperate tighter to calibrate the model to their own needs. We also plan to work on the tool support for the model, so that the model will be a part of a lager expert-based decision support system aimed to analyze, understand, manage and optimize a software development process.

ACKNOWLEDGMENTS

I would like to thank all participants of the questionnaire survey for providing useful information and additional thoughts inspiring this and further research. I would also like to thank Magne Jørgensen and Richard Torkar from Simula Research Laboratory in Norway and Norman Fenton from Queen Mary, University of London for valuable comments on building a Bayesian network model discussed here. This work has been supported by research funds from the Ministry of Science and Higher Education / National Science Centre in Poland as a research grant no. N N111 291738 for years 2010–2012.

REFERENCES

[1] Beaver JM. A life cycle software quality model using bayesian belief networks. Ph.D. Dissertation. University of Central Florida. Orlando; 2006.

[2] Catal C, Diri B. A systematic review of software fault prediction studies. Exp Syst Appl 2009, 36(4): 7346–7354.

[3] Fenton N, Marsh W, Neil M, Cates P, Forey S, Tailor M. Making Resource Decisions for Software Projects. In: Proceedings of the 26th International Conference on Software Engineering. Washington, DC: IEEE Computer Society; 2004, p. 397–406.

[4] Fenton NE, Neil M. A critique of software defect prediction models. IEEE Trans Softw Eng 1999; 25(5): 675–689.

[5] Fenton NE, Neil M, Caballero JG. Using Ranked Nodes to Model Qualitative Judgments in Bayesian Networks. IEEE Trans Knowl Data Eng 2007; 19(10): 1420–1432.

[6] ISO/IEC 25010:2011(E), Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models, 2011.

[7] Musa JD. Software Reliability Engineering: More Reliable Software Faster and Cheaper. Second Edition, Authorhouse; 2004.

[8] Nelson HJ, Poels G, Genero M, Piattini M. A conceptual modeling quality framework. Softw Qual J 2011; 20(1): 201–228.

[9] Radliński Ł. A conceptual Bayesian net model for integrated software quality prediction. Annales UMCS, Informatica 2011; 11(4): 49–60.

[10] Radliński Ł, A Framework for Integrated Software Quality Prediction using Bayesian Nets. In: Murgante B, Gervasi O, Iglesias A, Taniar D, Apduhan B, editors. Computational Science and Its Applications - ICCSA 2011. Lecture Notes in Computer Science; 6786, Berlin / Heidelberg: Springer; 2011, p. 310–325.

[11] Radliński Ł. Bayesian Network Model for Integrated Software Quality Prediction. 2012, http://lukrad.univ.szczecin.pl/projects/banisoq/.

[12] Radliński Ł. Empirical Analysis of the Impact of Requirements Engineering on Software Quality. In: Regnell B, Damian D, editors. Requirements Engineering: Foundation for Software Quality. Lecture Notes in Computer Science; 7195, Berlin / Heidelberg: Springer; 2012 p. 232–238.

[13] Radliński Ł. Enhancing Bayesian Network Model for Integrated Software Quality Prediction. In: Mauri JL, Lorenz P, editors. Proc. Fourth International Conference on Information, Process, and Knowledge Management, Valencia: IARIA; 2012, p. 144–149.

[14] Radliński Ł. Towards expert-based modeling of integrated software quality. J Theor Appl Comp Sci 2012 (under review).

[15] Riaz M, Mendes E, Tempero E. A systematic review of software maintainability prediction and metrics. In: Empirical Software Engineering and Measurement, Washington, DC: IEEE Computer Society; 2009, p. 367–377.

[16] Wagner S. A Bayesian network approach to assess and predict software quality using activity-based quality models. Inf Softw Technol 2010; 52(11): 1230-1241.

[17] Wagner S, Deissenboeck F. An Integrated Approach to Quality Modelling. In: Fifth International Workshop on Software Quality (WoSQ'07: ICSE Workshops 2007), Washington, DC: IEEE; 2007, p. 1.