Annales UMCS Informatica AI XI, 4 (2011) 49–60
DOI: 10.2478/v10065-011-0032-5

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI
http://www.annales.umcs.lublin.pl/

A conceptual Bayesian net model for integrated software quality prediction

Łukasz Radliński[1]*

[1] *Institute of Information Technology in Management, University of Szczecin*
Mickiewicza 64, 71-101 Szczecin, Poland

Abstract – Software quality can be described by a set of features, such as functionality, reliability, usability, efficiency, maintainability, portability and others. There are various models for software quality prediction developed in the past. Unfortunately, they typically focus on a single quality feature. The main goal of this study is to develop a predictive model that integrates several features of software quality, including relationships between them. This model is an expert-driven Bayesian net, which can be used in diverse analyses and simulations. The paper discusses model structure, behaviour, calibration and enhancement options as well as possible use in fields other than software engineering.

1 Introduction

Software quality has been one of the most widely studied areas of software engineering. One of the aspects of quality assurance is quality prediction. Several predictive models have been proposed since 1970's. A clear trade-off can be observed between model's analytical potential and the number of used quality features. Models that contain a wide range of quality features [**1, 2, 3**] typically have low analytical potential and are more frames for building calibrated predictive models. On the other hand, models with higher analytical potential typically focus on a single or very few aspects of quality, for example on reliability [**4, 5**].

This trade-off has been the main motivation for research focused on building predictive models that both incorporate various aspects of software quality and have high analytical potential. The aim of this paper is to build such predictive model as a

*lukrad@uoo.univ.szczecin.pl

Bayesian net (BN). This model may be used to deliver information for decision-makers about managing software projects to achieve specific targets for software quality.

Bayesian nets have been selected for this study for several reasons. The most important is related with the ability to incorporate both expert knowledge and empirical data. Typically, predictive models for software engineering are built using data-driven techniques like multiple regression, neural networks, nearest neighbours or decision trees. For the current type of study, a dataset with past projects of high volume and appropriate level of details is typically not available. Thus, the model has to be based more on expert knowledge and only partially on empirical data. Other advantages of BNs include the ability to incorporate causal relationships between variables, explicit incorporation of uncertainty through probabilistic definition of variables, no fixed lists of independent and dependent variables, running the model with incomplete data, forward and backward inference, and graphical representation. More information on the BN theory can be found in [**6, 7**] while recent applications in software engineering have been discussed in [**8, 9, 10, 11, 12, 13, 14, 15, 16**].

The rest of this paper is organized as follows: Section 2 brings closer the point of view on software quality that was the subject of the research. Section 3 discusses background knowledge used when building the predictive model. Section 4 provides the details on the structure of the proposed predictive model. Section 5 focuses on the behaviour of this model. Section 6 discusses possibilities for calibrating and extending the proposed model. Section 7 considers the use of such type of model in other areas. Section 8 summarizes this study.

2 Software Quality

Software quality is typically expressed in science and industry as a range of features rather than a single aggregated value. This study follows the ISO approach where software quality is defined as a "degree to which the software product satisfies stated and implied needs when used under specified conditions" [**1**]. This standard defines eleven characteristics, shown in Fig. 1 with dark backgrounds. The last three characteristics (on the left) refer to "quality in use" while others refer to internal and external metrics. Each characteristic is decomposed into the sub-characteristics, shown in Fig. 1 with white background. On the next level each sub-characteristic aggregates the values of metrics that describe the software product. The metrics are not shown here because they should be selected depending on the particular environment where such quality assessment would be used. Other quality models have been proposed in literature [**17, 3**] , from which some concepts may be adapted when building a customized predictive model.

In our approach we follow the general taxonomy of software quality proposed by ISO. However, our approach is not limited to the ISO point of view and may be adjusted according to specific needs. For this reason our approach uses slightly different

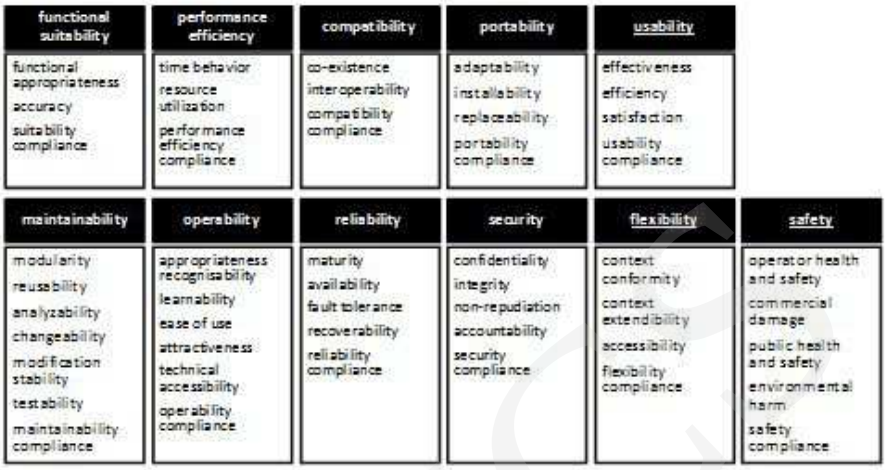

Fig. 1. Quality features and sub-features.

terminology with "features" at the highest level, "sub-features" at the second level and "measures" at the lowest level.

3 Background knowledge

Our approach assumes that the industrial-scale model for integrated software quality prediction has to be calibrated for specific needs and environment before it can be used in decision support. Normally such calibration should be performed among domain experts from the target environment, for example using a questionnaire survey. However, at this point such survey has not been completed yet so the current model has been built fully based on the available literature and expert knowledge of the modellers. This is the reason why the model is currently at the "conceptual" stage. The literature used includes quality standards [**1, 18, 2, 19, 20, 21**], widely accepted results on software quality [**22, 23, 24, 17, 3, 25, 26, 27, 28, 29**], and experience from building models for similar areas of software engineering [**8, 9, 10, 11, 12, 30, 13, 14, 15, 16**].

Available literature provides useful information on the relationships among quality features. Fig. 2 illustrates the relationships encoded in the proposed predictive model. There are two types of relationships: positive ("+") and negative ("–"). The positive relationship indicates a situation where the increased level of one feature causes a probable increase of the level of another feature. The negative relationship indicates a situation where the increased level of one feature causes a probably decrease of the level of another feature unless some compensation is provided. This compensation typically has a form of additional effort, increase of development process quality, or use of better tools or technologies.

Table 1 summarizes the relationships between the effort and the quality features.

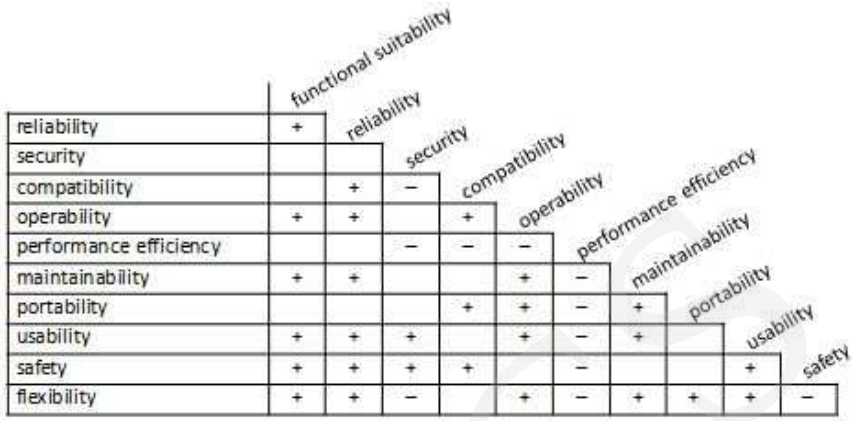

Fig. 2. Impact of controllable factors on quality features

Table 1. LMS types and features (Source: self study).

Quality feature	Requirements effort	Implementation effort	Testing effort
functional suitability	+	+	
reliability		+	+
performance efficiency		+	+
operability	+		
security			
compatibility			+
maintainability	+	+	
portability			
usability	+	+	+
flexibility	+	+	
safety			+

Currently there are two groups of controllable factors in the model: effort and process quality – defined separately for three development phases. It is assumed that the increase of effort or process quality has a positive impact on the selected quality features. This impact is not deterministic though, i.e. the increased effort does not guarantee better quality but causes that this better quality is more probable.

It should be noted that the relationships in Fig. 2 and Table 1 may be defined differently in specific target environments.

4 Model Structure

The proposed predictive model is a Bayesian net where the variables are defined as conditional probability distributions given their parents (i.e. immediate predecessors). It is beyond the scope of the paper to discuss the structure of the whole model because the full model contains over 100 variables. However, for full transparency and reproducibility of the results full model definition is available on-line [**31**].

Fig. 3 illustrates a part of the model structure by showing two quality features and relevant relationships. The whole model is a set of linked hierarchical naïve Bayesian classifiers where each quality feature is modelled by one classifier. Quality feature is the root of this classifier, sub-features are in the second level (children) and measures are the leaves.

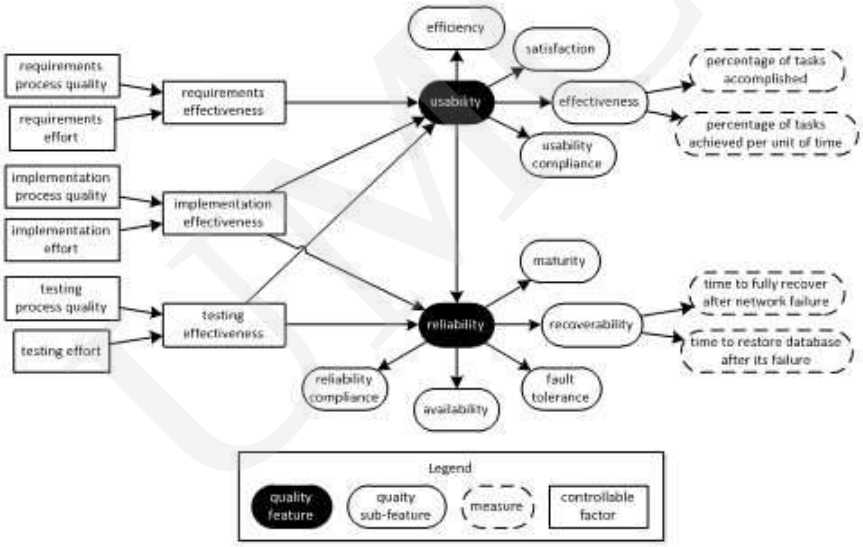

Fig. 3. Part of model structure.

To enable relatively easy model calibration and enhancement this model was built with the following assumptions:

- the links between various aspects of software quality may be defined only at the level of features;
- controllable factors are aggregated as the "effectiveness" variables, which, in turn, influence selected quality features.

Currently, all variables in the model, except measures, are expressed on a five-point ranked scale from 'very low' to 'very high'. Two important concepts, implemented in AgenaRisk tool [**32**], were used to simplify the definition of probability distributions. First, the whole scale of ranked variable is internally treated as the numeric a range (0, 1) with five intervals – i.e. for 'very low' an interval (0, 0.2), for 'low' an interval (0.2,

0.4) etc. This gives the possibility to express the variable not only as a probability distribution but also using summary statistics, such as the mean (used in the next section). It also opens the door for the second concept – using expressions to define probability distributions for variables. Instead of time-consuming and prone to inconsistencies manual filling probability tables for each variable, it is sufficient to provide only very few parameters for the expressions like Normal distribution (mean, variance), TNormal distribution (mean, variance, lower bound, upper bound), or weighted mean function – wmean(weight for parameter 1, parameter 1, weight for parameter 2, parameter 2 etc.). Table 2 provides the definitions for the selected variables in different layers of the model.

Table 2. Definition of selected variables.

Type	Variable	Definition
feature	usability	TNormal(wmean(1, 0.5, 3, wmean(3, *req_effect*, 2, *impl_effect*, 1, *test_effect*), 1, *funct_suit*), 0.05, 0, 1)
sub-feature	effectiveness	TNormal(*usability*, 0.01, 0, 1)
measure	percentage of tasks accomplished	*effectiveness* = 'very high' → Normal(95, 10) *effectiveness* = 'high' → Normal(90, 40) *effectiveness* = 'medium' → Normal(75, 60) *effectiveness* = 'low' → Normal(65, 80) *effectiveness* = 'very low' → Normal(50, 100)
controllable	testing effort	TNormal(0.5, 0.05, 0, 1)
controllable	testing effectiveness	TNormal(wmean(3, *test_effort*, 4, *test_procq*), 0.001, 0, 1)

5 Model Behaviour

To demonstrate model behaviour four simulations were performed with the focus to analyse the impact of one group of variables on another.

Simulation 1 was focused on the sensitivity analysis of quality features in response to the level of controllable factors. An observation about the state for a single controllable factor was entered to the model and then the predictions for all quality features were analyzed. This procedure was repeated for each state of each controllable factor.

Fig. 4 illustrates the results for one of such runs by demonstrating the changes of predicted levels of maintainability and performance efficiency caused by different levels of implementation effort. These results have been compared with the background knowledge in Table 1 to validate if the relationships have been correctly defined, i.e. if the change of the level of the controllable factor causes the assumed direction of changed level of quality feature. In this case the obtained results confirm that the background knowledge was correctly incorporated into the model.

With these graphs, it is possible to analyze the strength of impact of controllable factors on quality features. The impact of implementation effort is larger on maintainability than on performance efficiency – predicted probability distributions are more 'responsive' to different states of implementation effort for maintainability than for performance efficiency. Such information may be used in decision support.

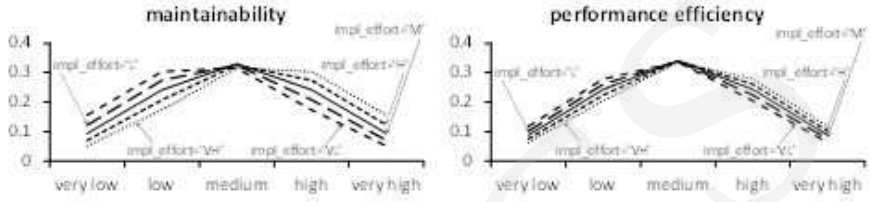

Fig. 4. Impact of implementation effort on the selected quality features.

Simulation 2 is similar to simulation 1 because it also analyses the impact of controllable factors on quality features. However, this simulation involves the analysis of summary statistics (mean values) rather than full probability distributions. Here, an observation 'very high' was entered to each controllable factor (one at the time) and then the mean value of predicted probability distribution for each quality feature was analyzed. Table 3 summarizes the results for effort at various phases. All of these mean values are above the default value of 0.5. These higher values suggest the increase in the predicted level specific quality features. These values correspond to "+" signs in Table 1 which further confirms the correct incorporation of the relationships between the controllable factors and the quality features.

Table 3. Predictions in simulation 2.

Quality feature	Requirements effort	Implementation effort	Testing effort
functional suitability	0.55	0.56	
reliability		0.56	0.55
performance efficiency		0.54	0.53
operability	0.60		
security			
compatibility			0.55
maintainability	0.56	0.57	
portability			
usability	0.56	0.55	0.52
flexibility	0.57	0.56	
safety			0.55

Simulation 3 was focused on the analysis of the relationships among various quality features. Similarly to simulation 2, it also covered the analysis of the mean values of predicted probability distributions. The results are presented in Table 4.

Table 4. Predictions in simulation 3.

Observed \ Predicted	functional suitability	reliability	security	compatibility	operability	performance efficiency	maintainability	portability	usability	safety	flexibility
functional suitability		0.55			0.55		0.58		0.58	0.53	0.57
reliability	0.55			0.55	0.52		0.55		0.55	0.54	0.54
security				0.46		0.46			0.53	0.52	0.47
compatibility		0.55	0.46		0.53	0.48		0.55		0.55	
operability	0.55	0.52		0.53		0.46	0.55	0.56	0.56		0.56
performance efficiency			0.46	0.48	0.46		0.47	0.44	0.48	<0.50	0.47
maintainability	0.57	0.55			0.55	0.47		0.56	0.57		0.58
portability				0.55	0.57	0.44	0.56				0.56
usability	0.58	0.55	0.53		0.56	0.48	0.58			0.54	0.57
safety	0.53	0.54	0.52	0.55		<0.50			0.54		0.48
flexibility	0.57	0.54	0.47		0.57	0.48	0.58	0.56	0.57	0.48	

The predicted mean values are either lower or higher than the default value 0.5. The values lower than 0.5 correspond to "–" signs in Fig. 2 while the values higher than 0.5 correspond to "+" signs in Fig. 2. Such results confirm that the model correctly incorporates the assumed relationships among quality features.

Simulation 4 has been focused on demonstrating more advanced model capabilities for delivering important information for decision support using what-if and trade-off analysis. Although such analysis may involve more variables, for simplicity, four variables were investigated: implementation effort, testing effort, maintainability, and performance efficiency. Some input data on the hypothetical project under consideration were entered into the model. The model provides predictions for these four variables as shown in Fig. 5 (scenario: baseline).

Let us assume that a manager is not satisfied with the low level of maintainability. Apart from previously entered input data, an additional constraint is entered to the model to analyze how to achieve high level of maintainability (*maintainability*='high'→ mean(*maintainability*)=0.7). As shown in Fig. 5, scenario: revision 1, the model predicts that such target is achievable with the increased level of implementation effort and testing effort (although the increase of required testing effort is very narrow). The model also predicts that the level of performance efficiency is expected to be lower. This is due to the negative relationship between the maintainability and performance efficiency (Fig. 2).

Let us further assume that, due to limited resources, not only the increase of effort is impossible, but even it has to be reduced to the level 'low' for implementation and

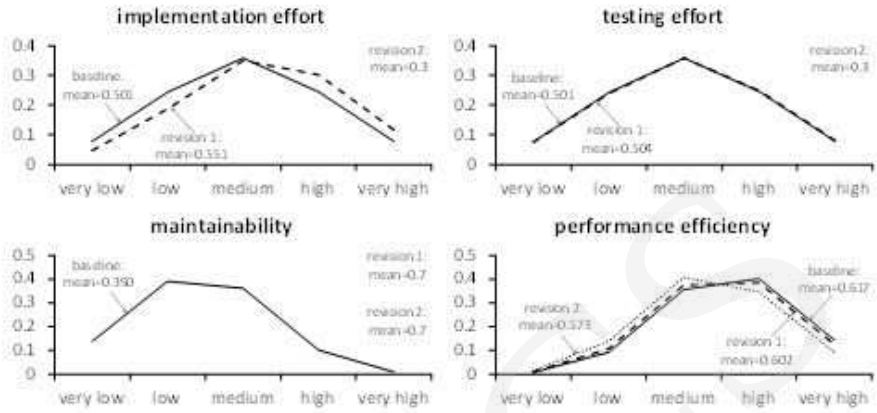

Fig. 5. Predictions in simulation 4.

testing. In such case the level of performance efficiency is expected to be further decreased (scenario: revision 2).

It is possible to perform various other types of simulations similar to simulation 4 to use the model with what-if, trade-off and goal-seeking analyses for decision support. Such simulation may involve more steps and more variables. Such simulations will be performed in future to enhance the validation of model correctness and usefulness.

6 Calibration and Enhancement Options

The proposed model has a structure that enables relatively easy calibration. As the variables are defined using expressions, the calibration requires setting appropriate parameters in these expressions:

- the values of weights in wmean functions – higher value for weight indicates stronger impact of particular variable on the aggregated value;
- the value of variance in TNormal expressions (second parameter) – value closer to zero indicates stronger relationship, higher values indicate lower relationships. Note, that since ranked variables are internally defined over the range (0, 1), typically a variance of 0.001 indicates very strong relationship and 0.01 – medium relationship.

Apart from calibration focused on the defining parameters for the existing structure, the model may be enhanced to meet specific needs:

- by adding new sub-features to features or new measures to sub-features – such change requires only the definition of newly added variable, no change in definitions of existing variables is necessary;

- by adding new controllable factors – such change requires the change in definition of "effectiveness" variable for specific phase, typically by setting new weights in wmean function;
- by adding new quality feature – such change requires the most work because it involves setting sub-features and measures, relationships among features, and relationships between the controllable factors and this new feature.

Currently the model does not contain many causal relationships. This may reduce the analytical potential. Defining the model using more causal relationships may increase analytical potential but may also make the model more difficult in calibration. Thus, this issue needs to be investigated carefully when building a tailored model.

The model enables static analysis, i.e. for the assumed point of time. Because both the project and the development environment evolve over time, it may be useful to reflect such dynamics in the model. However, such enhancement requires significantly more time spent on modelling and makes the calibration more difficult because more parameters need to be set.

7 Possible Use in Other Fields

The proposed predictive model is focused on the software quality area. Such approach may also be used in other fields/domains because the general constraints on model structure may also apply there. Possible use outside software quality area depends on the following conditions:

- the problem under investigation is complex but can be divided to a set of sub-problems,
- there is no or not enough empirical data to generate a reliable model from them,
- domain expert (or group of experts) is able to define, calibrate and enhance the model,
- the relationships are of stochastic and non-linear nature,
- there is a need for a high analytical potential.

However, even meeting these conditions, the use in other fields may be difficult. This happens in the case of a high number of additional deterministic relationships, which have to be reflected in the model with high precision. Possible use in other fields will be investigated in detail in future.

8 Conclusions

This paper introduced a new model for integrated software quality prediction. Formally, a model is a Bayesian net. This model contains a wide range of quality aspects (features, sub-features, measures) together with relationships among them. To make

the model useful in decision support it also contains a set of controllable factors (currently effort and process quality in different development phases).

This model encodes knowledge on software quality area published in literature as well as personal expert judgment. To prepare the model for using in the target environment it is necessary to calibrate the model, for example using questionnaires. The model may also be enhanced to meet specific needs. The model was partially validated for correctness and usefulness in providing information for decision support.

In future, such model may become a heart of an intelligent system for analysis and managing software quality. To achieve this higher level of automation would be required, for example in calibration and enhancement by automated extraction of relevant data/knowledge. In addition, the model would have to reflect more details on the development process, project or software architecture.

The stages of building customized models will be formalized in a framework supporting the proposed approach. This framework may also be used in building models with a similar general structure but in the fields other than software quality.

Acknowledgement: This work has been supported by the research funds from the Ministry of Science and Higher Education as a research grant no. N N111 291738 for the years 2010-2012.

References

[1] ISO/IEC 25000:2005, Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE (2005).
[2] ISO/IEC FDIS 9126-1:2001, Software Engineering - Product quality - Part 1: Quality model (2001).
[3] Kan S.H., Metrics and Models in Software Quality Engineering, Addison-Wesley, Boston (2003).
[4] Lyu M., Handbook of software reliability engineering, McGraw-Hill, Hightstown, NJ (1996).
[5] Musa J.D. Software Reliability Engineering: More Reliable Software Faster and Cheaper, Second Edition, Authorhouse (2004).
[6] Jensen F.V., Nielsen T.D., Bayesian Networks and Decision Graphs, Second Edition, Springer (2007).
[7] Kjærulff U.B., Madsen A.L., Bayesian Networks and Influence Diagrams, A Guide to Construction and Analysis, Springer, New York (2008).
[8] Abouelela M., Benedicenti L., Bayesian Network Based XP Process Modelling. International Journal of Software Engineering and Applications 1 (2010): 1.
[9] Beaver J.M., A life cycle software quality model using bayesian belief networks, Doctoral Dissertation, University of Central Florida, Orlando, FL (2006).
[10] Fenton N., Hearty P., Neil M., Radliński Ł., Software Project and Quality Modelling Using Bayesian Networks, In: Meziane, F., Vadera, S. (eds.) Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects, Information Science Reference, New York (2008): 1.
[11] Fenton N., Marsh W., Neil M., Cates P., Forey S., Tailor M., Making Resource Decisions for Software Projects, In: 26th Int. Conference on Software Engineering, Washington DC (2004): 397.

[12] Fenton N., Neil M., Marsh W., Hearty P., Radliński Ł., Krause P., On the effectiveness of early life cycle defect prediction with Bayesian Nets, Empirical Software Engineering 13 (2008): 499.
[13] Radliński Ł., Fenton N., Neil M., Marquez D., Improved Decision-Making for Software Managers Using Bayesian Networks, In: 11th IASTED Int. Conf. Software Engineering and Applications, IASTED, Cambridge, MA (2007): 13.
[14] Radlinski L., A Survey of Bayesian Net Models for Software Development Effort Prediction, International Journal of Software Engineering and Computing 2 (2010): 95 (in press).
[15] Van Koten C., Gray A.R., An application of Bayesian network for predicting object-oriented software maintainability, Information and Software Technology 48 (2006): 59.
[16] Wagner S., A Bayesian network approach to assess and predict software quality using activity-based quality models, In: 5th Int. Conf. on Predictor Models in Software Engineering, ACM Press, New York (2009).
[17] Jones C., Applied Software Measurement: Global Analysis of Productivity and Quality, Third Edition, McGraw-Hill, New York (2008).
[18] ISO/IEC FDIS 25010:2011, Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models (2011).
[19] ISO/IEC TR 9126-2:2003, Software engineering - Product quality - Part 2: External metrics (2003a).
[20] ISO/IEC TR 9126-3:2003, Software engineering - Product quality - Part 3: Internal metrics (2003).
[21] ISO/IEC TR 9126-4:2004, Software Engineering - Product quality - Part 4: Quality in use metrics (2004).
[22] Alvaro A., Santana De Almeida E., Romero De Lemos Meira S., A software component quality framework, ACM SIGSOFT Software Engineering Notes 35 (2010): 1.
[23] Côté M. A., Suryn W., Georgiadou E., In search for a widely applicable and accepted software quality model for software quality engineering, Software Quality Journal 15 (2007): 401.
[24] Jarvis A., Crandall V., Inroads to Software Quality: "How to" Guide and Toolkit, Prentice Hall PTR, Upper Saddle River, NJ (1997).
[25] O'Regan G., A Practical Approach to Software Quality, Springer-Verlag, New York (2002).
[26] Ortega M., Perez M., Rojas T., Construction of a Systemic Quality Model for Evaluating a Software Product, Software Quality Journal 11 (2003): 219.
[27] Rosqvist T., Koskela M., Harju H., Software Quality Evaluation Based on Expert Judgement, Software Quality Journal 11 (2003): 39.
[28] Schulmeyer G.G., McManus J.I. (eds.), Handbook of Software Quality Assurance, Prentice Hall PTR, Upper Saddle River, NJ (1999).
[29] Villalba M.T., Fernández-Sanz L., Martínez J.J., Empirical support for the generation of domain-oriented quality models, IET Software 4 (2010): 1.
[30] Fenton N.E., Neil M., Caballero J.G., Using Ranked Nodes to Model Qualitative Judgments in Bayesian Networks, IEEE Transactions on Knowledge and Data Engineering 19 (2007): 1420.
[31] Radliński Ł., BaNISoQ: Bayesian Net Model for Integrated Software Quality Prediction, http://lukrad.univ.szczecin.pl/projects/banisoq/ (2011).
[32] AgenaRisk BN Tool, Agena; http://www.agenarisk.com (2009).