Machine Learning Approach for Quality Assessment and Prediction in Large Software Organizations

Rakesh Rana, and Miroslaw Staron
Department of Computer Science and Engineering
Chalmers | University of Gothenburg
Göteborg, Sweden
Email: rakesh.rana@gu.se

***Abstract*—The importance of software in everyday products and services has been on constant rise and so is the complexity of software. In face of this rising complexity and our dependence on software - measuring, maintaining and increasing software quality is of critical importance. Software metrics provide a quantitative means to measure and thus control various attributes of software systems. In the paradigm of machine learning, software quality prediction can be cast as a classification or concept learning problem. In this paper we provide a general framework for applying machine learning approaches for assessment and prediction of software quality in large software organizations. Using ISO 15939 measurement information model we show how different software metrics can be used to build software quality model which can be used for quality assessment and prediction that satisfies the information need of these organizations with respect to quality. We also document how machine learning approaches can be effectively used for such evaluation.**

I. Introduction

Software affects almost every part of our life today, from consumer electronics we almost cannot do without to most services we consume on our computing devices and over internet are provided with help of software. Even the products such as cars which contained miniscule amount of software (if any) in 1970s today carry a large amount of software. With increasing importance of software in products and services we use and interact daily, a critical issue is of course system quality, attributes related to quality such as reliability, maintainability, re-usability etc. are of very high importance. Software quality, mainly the attributes related to dependability are even more important when developing software for systems deemed as safety, business and/or mission critical.

ISO 9126 [1] defines quality as *"the totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied needs"*. While ISO 25000 [2] takes the following approach to quality: *"capability of software product to satisfy stated and implied needs when used under specified conditions"*. Assessing software quality early in the development process are essential to identify and allocate resources where they are needed most [3].

Software metrics provide quantitative means to control software product and quality [4]. Software quality estimation models establishes relationships between desired software quality characteristics and measurable attributes. These models can be based on statistical techniques such as regression models [5] [6] or logical models [7] [8], since logical models such as those based on decision trees or rule sets are white-box models their interpretation is easier and thus also preferred [3].

Software metrics have long been used for monitoring and controlling software process [9][10], asses and/or improve software quality [11][12]. Metrics collection and analysis is part of daily work activities in large software development organizations [13]. Mature software development organizations also widely use the information model of ISO/IEC standard 15939 [14] as means of identifying the information needs and implementing measurement systems. Staron, Meding and Nilsson [13] provides a framework for quick development of measurement system in large software organizations using ISO/IEC 15939 standard. In this paper we propose how machine learning approaches can be used within the ISO/IEC 15939 information model framework for effective assessment and prediction of software quality. Framework that uses machine learning approaches within ISO/IEC 15939 information model will enhance the adoption of these techniques in large scale software organizations already using the standard for their information needs.

The rest of paper is organised as follows: In next section we describe very briefly software quality and ISO/IEC 15939 information model. Related work is discussed in section III, which is followed by section IV outlining the proposed framework. The paper is summarized in section V with conclusions and directions for future work.

II. Background

A. Software Quality

With increasing importance of software in our daily lives, the aspects of quality with respect to software have also gained high importance. As with many attributes, quality can be improved effectively if we define it properly and measure it continuously.

While quality is one of the very common and well known terms, yet it is ambiguous and also commonly misunderstood. To many people, quality is similar to what a federal judge once said about obscenity *"I know it when I see it"* [15]. The main reasons for ambiguity and confusion can be attributed to the fact that quality is not a single idea, but a multidimensional concept, where dimensions includes the entity of interest, the viewpoint and the attributes of that entity [15]. Thus, to fully appreciate the complexities related to quality the shift have been from defining quality from a single perspective towards defining and working with quality models. Quality model

according to ISO/IEC 25000 [2] is: *"defined set of characteristics, and of relationships between them, which provides a framework for specifying quality requirements and evaluating quality"*.

The latest and now widely used framework for software quality is given in SQuaRE: Software product Quality Requirements and Evaluation, ISO/IEC 25000 [2] which provides a series of standards on product quality requirements and evaluation. Quality model division in SQuaRE series (2501n) provides the detailed software quality model that includes characteristics for internal, external and quality in use. Quality characteristics are further decomposed into subcharacteristics and practical guidance for their use is also provided. Table I & II outlines the characterstics and subcharacteristics of ISO/IEC 9126-1 [1] quality model.

TABLE I. ISO/IEC 9126-1 INTERNAL/EXTERNAL QUALITY MODEL CHARACTERISTICS AND SUBCHARACTERISTICS.

Characteristics	Subcharacteristcs
Functionality	suitability
	accuracy
	interoperability
	security
	functionality compliance
Reliability	maturity
	fault tolerance
	recoverability
	reliability compliance
Usability	understandability
	learnability
	operability
	attractiveness
	usability compliance
Efficiency	time behaviour
	resource efficiency utilisation
	efficiency compliance
Maintainability	analysability
	changeability
	stability
	testability
	maintainability compliance
Portability	adaptability
	installability
	co-existence
	replaceability
	portability compliance

TABLE II. QUALITY IN USE ACCORDING TO ISO/IES 9126-1.

Measure	Characteristics
Quality in Use	Effectiveness
	Productivity
	Safety
	Satisfaction

Depending on the product, the way and/or the environment in which it would be used, different quality measures, characteristics and subcharacteristics can be weighted differently to customize the quality model to specific needs.

B. ISO 15939 Measurement Information Model

ISO/IEC standard 15939 [14] defines the activities and tasks necessary to implement a measurement process. The measurement information model provide a structure that links information need of stakeholders to the measurable attributes. The measurement information model outlines how relevant attributes are qualified and converted to indicators that supplys the information needed for decision making. The overview of model as given in [16] and measurement information model adapted from ISO/IEC 15939 are presented in figure 1.

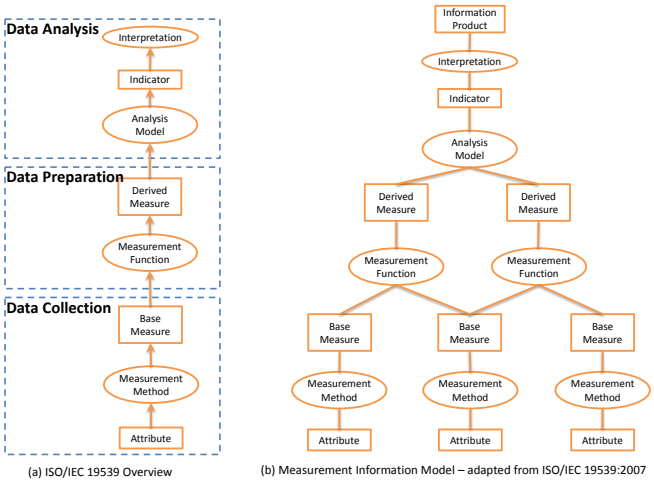

Fig. 1. Overview & Measurement information model adapted from ISO/IEC 15939: 2007

For detailed description of measurement information model and implementing a measurement process, readers are referred to standard ISO/IEC 15939 [14]. Two key components of the information model we would emphasise on this paper are provided here with their definitions from the standard:

Measurement Function: *A function is an algorithm or calculation performed to combine two or more base measures. The scale and unit of derived measure depend on the scales and units of the base measures from which it is composed as well as how they are combined by the function*

(Analysis) Model: *An algorithm or calculation combining one or more base and/or derived measures with associated decision criteria. It is based on an understanding of, or assumptions about, the expected relationship between the component measures and/or their behaviour over time. Models produce estimates or evaluations relevant to defined information needs. The scale and measurement method affect the choice of analysis techniques or models used to produce indicators.*

III. RELATED WORK AND DISCUSSION

The application of Machine Learning (ML) techniques in the software engineering domain has been increasing rapidly. Zhang and Tsai [17] provide a detailed overview of machine learning approaches applied to range of software engineering problems. A number of recent studies have looked at applying different machine learning techniques for assessment and predictions of various aspects of software quality, Myra [18] uses ML to support quality judgements, while Azar et al. [3] proposes search based hybrid heuristic approach to optimize rule-based software quality assessment. ML approaches have also been used to model and predict software ageing in study by Andrzejak and Silva [19] and finding latent code errors by Brun and Ernst [20].

With regard to software quality assessment and prediction using ML approaches, much effort has gone into identifying and predicting defect prone software modules. Khoshgoftaar, Allen and Deng [21] used regression trees to classify fault prone software modules, while in different studies Fenton [22] demonstrated usefulness of Bayesian nets for predicting software defects. [23] uses support vector machines for predicting modules that are defect-prone. Support Vector Machines (SVM) are also used and compared to Artificial Neural Networks (ANN) in work by Iker Gondra [24] which also looks at determining the importance of each software metric in predicting fault-proneness. Various ML techniques for predicting defect prone software modules are assessed and compared in work by Khoshgoftaar and Seliya [25].

Most of the previous studies using ML approaches for software quality assessment and prediction have been focused on one or few aspects of software quality such as defect proneness or reliability. Using ML for assessing or predicting holistic software quality are rare which is not surprising as software quality is a multi-dimensional concept and difficult to define. Almeida and Matwin [7] describes machine learning approach for building software quality models casting quality prediction as concept learning problem.

Software quality models such as one proposed by Bohem et al. [26] and international standards on software quality [1], [2] have greatly enhanced our understanding of what factors affects quality, they have also contributed towards standardization of various terms and their definitions with regard to quality. But as described by Kitchenham and Pfleeger [27], following limitations still holds true with respect to software quality models:

- Software quality models lack the rationale for determining which factors to include or exclude when defining quality,
- The basis of selection of quality characteristics and subcharacteristics and their order is unclear and different in different quality models, and
- Software quality models in general lacks the explanation of how lowest-level metrics (referred as indicators in ISO 9126) are composed to form higher level quality characteristics.

The authors argues that the absence of explanation of how higher level quality characteristics are related to lower level factors/metrics (or indicators) imply that there are no means to measure factors at top of hierarchy which makes the model untestable.

Geoff Dromey [28] believes that these quality models are not much useful for software engineers looking to build quality product. The reasoning is that building high level quality attributes such as reliability or maintainability into products is almost impossible using such models. Dromey's proposed approach [28] overcomes some of these limitations, the proposed framework suggests identifying quality attributes, components, quality-carrying properties and linking these properties to quality attributes which provides reference against which quality can be assessed for individual module, components or even at system level.

Our framework compliments both the software quality models such as ISO 9126 and the approach suggested by Geoff Dromey. For quick and effective assessment of software quality in large organizations we propose using machine learning techniques integrated within the measurement information model of ISO/IEC 15939.

IV. Framework for Quality Assessment and Prediction using Machine Learning Techniques

As explained earlier large mature software development organizations usually collects and monitor various software metrics considered important for the purpose of monitoring and controlling software development process and software/product quality. Given the availability of this large set of data for current as well as historical projects and the unclarity of how low level software metrics affect high order quality characteristics (or overall quality), *we contend that for effective assessment and prediction of overall software quality in large organizations, machine learning techniques such as pattern recognition and classification can be used efficiently*. We also provide a framework for quality assessment and prediction which integrates ML techniques in framework of measurement information model of ISO/IEC 15939 - a widely used standard to develop and implement measurement systems within the software industry.

In the framework we first take a bottoms-up approach, given that we have some quantitative assessment of high order quality characteristics (as per software quality models by Bohem [29] or ISO 9126 [1]), we can use ML techniques for pattern recognition such as Artificial Neural Networks (ANN) to recognise/predict under which quality category a given software module/product falls at a given point of time during its development. The model for such assessment/prediction can be represented as in figure 2.

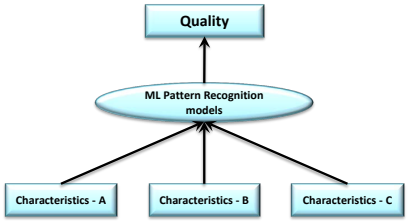

Fig. 2. Overview of quality assessment model using ML pattern recognition

Next to get a quantitative assessment (quantitative assessment here also refer to value on a limited scale, say between 0-10) or qualitative class labels (such as high/low) values for different high order quality characteristics that affect overall software quality, we propose using framework of measurement information model of ISO/IEC 15939 supported by the machine learning techniques of pattern recognition and classification. This model would be very much based on ISO/IEC 15939 measurement information model with a top-down approach which is represented in figure 3.

The model to asses the individual quality characteristics can be obtained using top-down approach as in ISO/IEC 15939 measurement information model, following steps would be involved:

- First depending on the characteristics of given software project/product and needs of different stake-

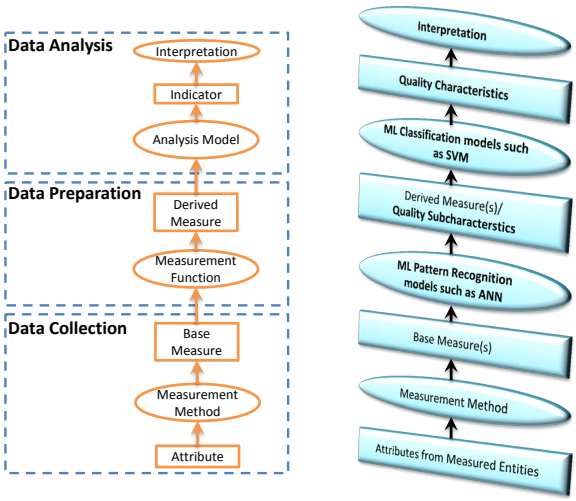

Fig. 3. Overview of high order quality characteristics assessment model using ML techniques

holders, information need is established. The only difference compared to ISO/IEC 15939 measurement information model here is that information need here specifically relates to quality characteristics that is recognised as important for overall quality of given software project/product.

- Next for identified information need (quality characteristics), subcharacteristics (corresponding to derived measures in reference to ISO/IEC 15939) and different attributes/software metrics that can potentially affect the given subcharacteristics are identified.

- The next step is data collection which includes collection of attributes and using measurement theory to assign them values to obtain the relevant base measures. This step also remains unchanged in our framework with respect to ISO/IEC 15939 measurement information model.

- Different base measure(s) can now be combined to form derived measures using pattern recognition techniques (e.g. Artificial Neural Networks) from the machine learning toolbox. The main advantage of using ML techniques in this step is that using historical data, we can easily and effectively use the pattern recognition ability of ML approaches while finding formal mathematical relations for the same is complex and difficult.

- After obtaining the quality subcharacteristics (derived measures) we can again use the machine learning techniques such as classification models (e.g. Support Vector Machines) which can use the historical data to classify given software project/product/module to a class of quality characteristics. Again ML learning tools are highly useful in this step as finding the correct analysis model is difficult and complex.

- The obtained quality characteristics for current software project/product/module can then be interpreted. The interpretation of quality characteristics in this framework is comparatively easier given that using ML model based on historical data, the obtained values are compared in light of earlier experience within the organization.

Thus by following the above steps and using machine learning techniques in conjunction to ISO/IEC 15939 measurement information model we can model overall quality of given software module/product/project. We can also effectively used the collected software metrics and historical data which are usually widely available within large software development organizations.

V. Conclusion

In this paper we examined the growing importance of software in products and services we consume regularly. Given this high and growing dependence on software its quality is of course a major concern for us as consumers and software development organizations who develop these products/services. As software becomes more integral part of our daily lives and its complexity increases - monitoring, assessing and improving software quality also becomes ever more important.

Quality in context of software is a common yet ambiguous term. While everyone have a feeling for quality, it's an complex and multifaceted concept, it varies based on the perspective and also on the environment in which a product is used and user expectations. While software quality models and international standards have helped us understand better the factors that may affect quality, the relationship and effect size of individual factors/subcharacteristics on overall quality is unknown. Finding precise relationships is not only difficult but may well be impossible given that quality depends on number of characteristics internal, external and in-use which in-turn can be affected by very large number of factors.

Large and mature software development organizations collects and monitors software metrics widely and is a part of their day today activities for all projects. While this data is rarely released outside these organizations due to confidentiality issues, this wealth of data can be effectively used internally to model software quality using traditional and machine learning based techniques.

Given the high availability of wide range of software metrics data for historical and undergoing projects and use of ISO/IEC 15939 for implementing measurement systems in large software organizations, we proposed a framework that uses machine learning techniques in conjunction with measurement information model of ISO/IEC 15939. Using machine learning approaches means that we no longer need to know exact relationships between base and derived measures and build precise analysis model of how different quality subcharacteristics affect higher order quality characteristics or overall quality. Using the historical data machine learning techniques can help assess overall quality and high order quality characteristics models based on measurable attributes. Another very important benefit of using machine learning techniques is that they are self improving, thus as they are used in these large organizations and more data is collected over time, their accuracy and predictive power improves making them very attractive for such analysis.

While machine learning approaches have been applied to many software engineering problems and also to many individual software quality characteristics/subcharacteristics, their use for overall quality assessment and prediction is rare. The framework presented in this paper need to be validated in a large software organization setting which we see as our future work direction. We also believe that more research is needed in this area to establish models for evaluating and predicting higher order quality characteristics and overall quality using widely available software metrics data using machine learning techniques.

ACKNOWLEDGMENT

The work has been funded by Vinnova and Volvo Cars jointly under the FFI programme (VISEE, Project No: DIARIENR: 2011-04438).

REFERENCES

[1] ISO, *ISO Standard 9126: Software Engineering Product Quality, parts 1, 2 and 3*, Std., 2001.

[2] ——, *International Standard. Software Engineering Software product Quality Requirements and Evaluation (SQuaRE)*, Std., 2005.

[3] D. Azar, H. Harmanani, and R. Korkmaz, "A hybrid heuristic approach to optimize rule-based software quality estimation models," *Information and Software Technology*, vol. 51, no. 9, pp. 1365–1376, 2009.

[4] H. Lounis and L. Ait-Mehedine, "Machine-learning techniques for software product quality assessment," in *Quality Software, 2004. QSIC 2004. Proceedings. Fourth International Conference on*. IEEE, 2004, pp. 102–109.

[5] F. Brito e Abreu and W. Melo, "Evaluating the impact of object-oriented design on software quality," in *Software Metrics Symposium, 1996., Proceedings of the 3rd International*. IEEE, 1996, pp. 90–99.

[6] V. R. Basili, L. C. Briand, and W. L. Melo, "A validation of object-oriented design metrics as quality indicators," *Software Engineering, IEEE Transactions on*, vol. 22, no. 10, pp. 751–761, 1996.

[7] M. A. De Almeida and S. Matwin, "Machine learning method for software quality model building," in *Foundations of Intelligent Systems*. Springer, 1999, pp. 565–573.

[8] S. Zhong, T. M. Khoshgoftaar, and N. Seliya, "Analyzing software measurement data with clustering techniques," *Intelligent Systems, IEEE*, vol. 19, no. 2, pp. 20–27, 2004.

[9] M. Staron and W. Meding, "Monitoring bottlenecks in agile and lean software development projects–a method and its industrial use," in *Product-Focused Software Process Improvement*. Springer, 2011, pp. 3–16.

[10] N. Mellegård and M. Staron, "Characterizing model usage in embedded software engineering: a case study," in *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*. ACM, 2010, pp. 245–252.

[11] M. Staron, L. Kuzniarz, and C. Thurn, "An empirical assessment of using stereotypes to improve reading techniques in software inspections," in *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4. ACM, 2005, pp. 1–7.

[12] N. Mellegard, M. Staron, and F. Torner, "A light-weight defect classification scheme for embedded automotive software and its initial evaluation," in *Software Reliability Engineering (ISSRE), 2012 IEEE 23rd International Symposium on*. IEEE, 2012, pp. 261–270.

[13] M. Staron, W. Meding, and C. Nilsson, "A framework for developing measurement systems and its industrial evaluation," *Information and Software Technology*, vol. 51, no. 4, pp. 721–737, 2009.

[14] ISO/IEC, *Systems and software engineering – Measurement Process*, Std., 2007.

[15] S. H. Kan *et al.*, *Metrics and Models in Software Quality Engineering, 2/e*. Pearson Education India, 2003.

[16] L. Bégnoche, A. Abran, and L. Buglione, "A measurement approach integrating iso 15939, cmmi and the isbsg," in *Proceedings of 4th Software Measurement European Forum (SMEF), Rome*, 2007.

[17] D. Zhang and J. J. Tsai, "Machine learning and software engineering," *Software Quality Journal*, vol. 11, no. 2, pp. 87–119, 2003.

[18] M. Custard and T. Sumner, "Using machine learning to support quality judgments," *D-Lib Magazine*, vol. 11, no. 10, pp. 1082–9873, 2005.

[19] A. Andrzejak and L. Silva, "Using machine learning for non-intrusive modeling and prediction of software aging," in *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*. IEEE, 2008, pp. 25–32.

[20] Y. Brun and M. D. Ernst, "Finding latent code errors via machine learning over program executions," in *Proceedings of the 26th International Conference on Software Engineering*. IEEE Computer Society, 2004, pp. 480–490.

[21] T. M. Khoshgoftaar, E. B. Allen, and J. Deng, "Using regression trees to classify fault-prone software modules," *Reliability, IEEE Transactions on*, vol. 51, no. 4, pp. 455–462, 2002.

[22] N. Fenton, M. Neil, W. Marsh, P. Hearty, D. Marquez, P. Krause, and R. Mishra, "Predicting software defects in varying development lifecycles using bayesian nets," *Information and Software Technology*, vol. 49, no. 1, pp. 32–43, 2007.

[23] K. O. Elish and M. O. Elish, "Predicting defect-prone software modules using support vector machines," *Journal of Systems and Software*, vol. 81, no. 5, pp. 649–660, 2008.

[24] I. Gondra, "Applying machine learning to software fault-proneness prediction," *Journal of Systems and Software*, vol. 81, no. 2, pp. 186–195, 2008.

[25] T. M. Khoshgoftaar and N. Seliya, "Comparative assessment of software quality classification techniques: An empirical case study," *Empirical Software Engineering*, vol. 9, no. 3, pp. 229–257, 2004.

[26] B. Boehm, J. Brown, H. Kaspar, M. Lipow, G. MacLeod, and M. Merritt, "Characteristics of software quality. 1978."

[27] B. Kitchenham and S. L. Pfleeger, "Software quality: the elusive target [special issues section]," *Software, IEEE*, vol. 13, no. 1, pp. 12–21, 1996.

[28] R. G. Dromey, "Cornering the chimera [software quality]," *Software, IEEE*, vol. 13, no. 1, pp. 33–43, 1996.

[29] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, 1988.