Poster: Introducing Quality Models Based On Joint Probabilities

Maria Ulan, Welf Löwe, Morgan Ericsson, Anna Wingkvist
Linnaeus University
Växjö, Sweden
{maria.ulan|welf.lowe|morgan.ericsson|anna.wingkvist}@lnu.se

ABSTRACT

Multi-dimensional goals can be formalized in so-called *quality models*. Often, each dimension is assessed with a set of *metrics* that are not comparable; they come with different units, scale types, and distributions of values. Aggregating the metrics to a single quality score in an ad-hoc manner cannot be expected to provide a reliable basis for decision making. Therefore, aggregation needs to be mathematically well-defined and interpretable. We present such a way of defining quality models based on *joint probabilities*. We exemplify our approach using a quality model with 30 standard metrics assessing technical documentation quality and study ca. 20,000 real-world files. We study the effect of several tests on the independence and results show that metrics are, in general, not independent. Finally, we exemplify our suggested definition of quality models in this domain.

CCS CONCEPTS

• **Social and professional topics** → *Quality assurance*;

KEYWORDS

Quality assessment, Software metrics, Bayesian networks

ACM Reference Format:
Maria Ulan, Welf Löwe, Morgan Ericsson, Anna Wingkvist. 2018. Poster: Introducing Quality Models Based On Joint Probabilities. In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3195103

1 INTRODUCTION

The ISO/IEC 25010:2011 standard [3] describes relationships between software attributes and qualities by a so-called Software Quality Model. It follows the Factor-Criteria-Metric-structure [4], where quality characteristics are defined in terms of sub-characteristics in a tree-like structure. In general, a specific metric does not measure a quality characteristic directly, so we need to combine different metrics to measure a single (sub-)characteristic. Quality models provide a basic understanding what data to collect and which metrics to use. However, it is unclear how the metrics should be aggregated to a single score for a (sub-)characteristic.

Metrics have different units, scale types, and distributions of values, which makes quality assessment challenging [1]. The distribution of values is usually skewed, so well-known aggregation methods, such as *mean* and *median*, should not be used. There are efforts to avoid these problems: [7] suggest the use of inequality indices, e.g., the Gini index, and the SQUALE model [5] (and many others) relies on non-parametric statistics. In both these cases, there is no aggregation to a total quality score; something we require. The Gini index cannot identify the unequal part of the distribution and aggregation is only performed on metrics level. The SQUALE model focuses on detecting bad quality and gives no impression of the overall quality.

The aggregation method should provide a sound basis for decision making, which ad-hoc methods do not. Decisions should be aligned with intuitive reasoning, expert knowledge, and common sense based on mathematical grounds to remove uncertainty and subjectivity. To address this, we suggest an automated aggregation approach, where quality scores are as joint probabilities.

2 APPROACH

By definition, each metric is a function and can be treated as a random variable. We assume that values for each metrics are known and without loss of generality, all metrics are defined such that larger values indicate worse quality. We calculate the *complementary Cumulative Distribution Function* ($1 - CDF$), and define a *metric's score* as the probability of finding another entity with a metric value greater than or equal to the given value. We can now consider the joint probability distribution of several metrics, and each criterion can be expressed as the joint $1 - CDF$ of all metrics that contribute to the criterion. The factors are expressed as the joint $1 - CDF$ of their criteria. The aggregation, and quality model in extension, expresses quality as the probability of observing something with equal or worse quality, based on all software projects observed; good and bad quality is expressed in terms of lower and higher probabilities. The quality and its interpretation is the same on all levels of the quality models, from metrics scores to factors. To calculate the joint probability for several metrics, we could use the classical *Chain rule*. However, if we assume that the domain size of the metrics is s, both time and space complexity are $O(s^n)$. We can reduce the number of operations required by studying dependencies between metrics. A *Bayesian Network (BN)* [6] represents a probability distribution factorized along a *Directed Acyclic Graph (DAG)* where nodes represent events and edges represent dependencies. We model events related to metrics as nodes. The criteria can then be quantified as joint probabilities, and the network shows how these can be calculated efficiently. Statistical analysis of a sample of the actual (metrics) data implies the structure of the network. Once the structure is learned, a quality model can be defined in an understandable, interpretable, actionable, and mathematically

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195103

sound way; each criteria is defined as joint probability of metrics. A criteria can be considered as an indirect metric, so factors can be modeled the same way.

3 RESULTS

We conducted an experiment to evaluate the proposed approach for quality assessment. We relied on a real-world case where a metrics-based quality model is used to evaluate (systems) documentations. The quality model relies on 31 metrics that are used to evaluate 8 criteria: Cloning issues, Anti-patterns, File complexity, Hierarchy complexity, Language issues, Referential complexity, Text complexity, and Validity issues. The data set evaluated was composed of 116 documentations consisting of a total of 21,121 files. The samples for testing were based on 33 evaluations; a set of 16 metrics were used in 20 evaluations for Document Information Typing Architecture (DITA) structured documentations and a set of 31 metrics were used in 13 evaluations for XML structured documentations.

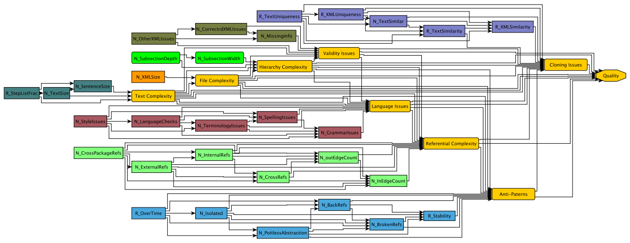
Dependencies. Independence tests of random variables depends on their distribution and the type of relationship. We applied a number of such tests to the metrics with significance levels 0.01 and 0.05 (cf. Table 1) and found that they cannot be treated as independent variables.

Table 1: Independence Tests

Independence test	Number of independent pairs			
	α = 0.01		α = 0.05	
	DITA	XML	DITA	XML
spearman	11	54	6	39
kendall	9	53	0	14
mic	8	42	5	42
hoeffd	5	120	5	106
genest	14	65	10	48
lis	0	0	0	0
Total number of pairs	120	465	120	465

Aggregation. To construct a *BN*, we need to determine variables that should be modeled, represent these as nodes, decide which nodes should be linked, and construct conditional probability tables for each node. The learning uses the best dependency detection algorithm, chosen from following alternatives. From the set of constraint-based algorithms, we consider *Interleaved Incremental Association* with significance levels 0.01 and 0.05 to avoid false positives during the Markov blanket detection process. From the score-based algorithms, we consider *Hill-Climbing* with *Bayesian Information Criterion* and *Log-likelihood* scores. From the hybrid algorithms, we considered the more general *2-phase Restricted Maximization* algorithm with significance levels 0.01 and 0.05, and the *Log-likelihood* score. For both DITA and XML data, the score-based algorithm showed the best results. The constrained-based algorithm implementation resulted in undirected graphs, i.e., the conditional tests could not always define the direction of edges and the hybrid algorithm did not represent all dependencies. We constructed 10,000 networks during the learning with retraining and the strength of every possible edge was calculated using the bootstrap approach [2]. We use the *Hill-Climbing* algorithm with *Log-likelihood* score. Edges which appear in a minimum of 95% of the networks were retrained and a directed acyclic graph was constructed based on the retrained edges. The quality model is represented as a *DAG* whose nodes correspond to quality characteristics while edges represent dependencies, cf. Figure 1.

Figure 1: Formal Quality Model



4 CONCLUSION AND FUTURE WORK

Domain experts can easily define an initial intuitive quality model. However, different metrics measure different aspects of quality and our study confirms that metrics are in general not independent. We show that a well-defined and interpretable quality model can be automatically constructed based on an initial intuitive model and sample metrics data. The proposed model has an advantage over deterministic models: it can be empirically validated in a probabilistic sense. This model can then be used to assess the quality of yet unknown artifacts. Learning the Bayesian networks is computationally expensive since it involves both NP-complete and NP-hard problems. Future work will trade the accuracy of the learning approaches for efficiency and evaluate the effects. The proposed approach and many of the metrics could easily be applied to an arbitrary software project. We will evaluate the effectiveness of the approach using other cases. We assume that larger values indicate worse quality in this paper, but this is not a limitation since we can transform metrics values to have this property. We do not consider aggregation of scores along the structure of the artifacts, e.g., from classes to packages. We are convinced that our approach supports this in combination with the aggregation we presented, but we need to show and evaluate it using real-world data.

REFERENCES

[1] M. Ericsson, W. Löwe, T. Olsson, D. Toll, and A. Wingkvist. 2013. A Study of the Effect of Data Normalization on Software and Information Quality Assessment. In *20th Asia-Pacific Software Engineering Conf. (APSEC)*, Vol. 2. 55–60.

[2] N. Friedman, M. Goldszmidt, and A. Wyner. 1999. Data Analysis with Bayesian Networks: A Bootstrap Approach. In *Proc. of the 15th Conf. on Uncertainty in Artificial Intelligence (UAI)*. Morgan Kaufmann, 196–205.

[3] ISO/IEC. 2010. *ISO/IEC 25010 System and software quality models.* Technical Report.

[4] J. McCall, P. Richards, and G. Walters. 1977. *Factors in software quality: Final report.* General Electric Company.

[5] K. Mordal-Manet, F. Balmas, S. Denier, S. Ducasse, H. Wertz, J. Laval, F. Bellingard, and P. Vaillergues. 2009. The squale model; A practice-based industrial quality model. In *2009 IEEE Int. Conf. on Software Maintenance (ICSM)*. 531–534.

[6] J. Pearl. 1982. *Reverend Bayes on inference engines: A distributed hierarchical approach.* Cognitive Systems Laboratory, School of Engineering and Applied Science, University of California, Los Angeles.

[7] B. Vasilescu, A. Serebrenik, and M. van den Brand. 2011. You can't control the unfamiliar: A study on the relations between aggregation techniques for software metrics. In *27th IEEE Int. Conf. on Software Maintenance (ICSM)*. 313–322.