A Bayesian Network Approach to Assess and Predict Software Quality Using Activity-Based Quality Models

Stefan Wagner
Fakultät für Informatik
Technische Universität München
Garching b. München, Germany
wagnerst@in.tum.de

ABSTRACT

Assessing and predicting the complex concept of software quality is still challenging in practice as well as research. Activity-based quality models break down this complex concept into more concrete definitions, more precisely facts about the system, process and environment and their impact on activities performed on and with the system. However, these models lack an operationalisation that allows to use them in assessment and prediction of quality. Bayesian Networks (BN) have been shown to be a viable means for assessment and prediction incorporating variables with uncertainty. This paper describes how activity-based quality models can be used to derive BN models for quality assessment and prediction. The proposed approach is demonstrated in a proof of concept using publicly available data.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics; D.2.9 [**Software Engineering**]: Management—*Software Quality Assurance (SQA)*

General Terms

Measurement, Management

Keywords

Activity-Based Quality Models, Bayesian Networks, Quality Assessment, Quality Prediction

1. INTRODUCTION

Despite the importance of software quality, the management of software quality is still an immature discipline in software engineering research and practice. Research work has gone in many directions and produced a variety of interesting results. However, there is still no commonly agreed way for quality management. The practice varies strongly from a concentration on testing to a large-scale quality management process.

One main problem is that many of the tools and methods work rather isolated. Hence, quality is tackled on many levels without a combined strategy [16]. What is missing is a clear integration of these single efforts. One prerequisite for such an integration is a quality management sub-process in the overall development process. The process defines the roles, activities and artefacts and how these work together. A second prerequisite is a clear quality model that defines quality for the software to be developed. The term *quality model* is used in various contexts but we use it here to denote a model that breaks down the complex concept *quality* and thereby makes it more concrete. These concrete descriptions of quality could then be used to construct, assess and predict software quality.

1.1 Problem

Current quality models such as the ISO 9126 [10] have widely acknowledged problems [6, 11]. Especially as a basis for assessment and prediction, the defined "-ilities" are too abstract. A clear transition to measurements is therefore difficult in practice. Hence, quantitative quality assessment and prediction is usually done without direct use of such a quality model. This, in turn, leads to isolated solutions in quality management.

1.2 Contribution

We use the previously proposed activity-based quality models (ABQM) as a basis for quality assessment and prediction. They provide a clear structure of quality and detailed information about quality-influences. Activity-based quality models have proven useful in practice to structure quality and to generate corresponding guidelines and checklists. In this paper, we add a systematic transition from ABQMs to Bayesian networks in order to enhance their assessment and prediction capabilities. A 4-step approach is defined that generates a Bayesian network using an activity-based quality model and an assessment or prediction goal. The approach is demonstrated in a proof of concept.

1.3 Outline

We first motivate and introduce activity-based quality models in Section 2. In Section 3 the 4-step approach for systematically constructing a Bayesian network from an activity-based quality model is proposed. The approach is then demonstrated in a proof of concept in Section 4 using publicly available data from a NASA project. Related work is discussed in Section 5 and final conclusions are given in Section 6,

2. ACTIVITY-BASED QUALITY MODELS

Quality models describe in a structured way the meaning of a software's quality. We introduce the use of general quality models and how the modelling of activities and facts helps to define quality more precisely.

2.1 Software Quality Models

If quality requirements are handled at all in a software project, quality models will be an integral part of it. The quality model describes what is meant by *quality* and refines this concept in a structured way. In practice, this is often reduced to metrics such as *number of defects* or high-level descriptions as given by the ISO 9126 [10].

In general, there are two main uses of quality models in a software project: (1) as a basis for defining quality requirements and (2) for relating quality assurance techniques and measurements to the quality requirements. The first is commonly done by constraining well-known quality attributes (reliability, maintainability, ...) as defined in a quality model. In practice, this is often reduced to simple statements such as "The system shall be easily maintainable." The second use is often not explicitly considered. However, certain metrics are nearly always measured such as the number of faults in the system detected by inspections and testing. The relationship between these measures and quality attributes remains unclear. The reason lies in the lack of practical means to define metrics for these high-level quality attributes. Hence, more structure and more detail is needed in quality models to integrate them closely in the development process.

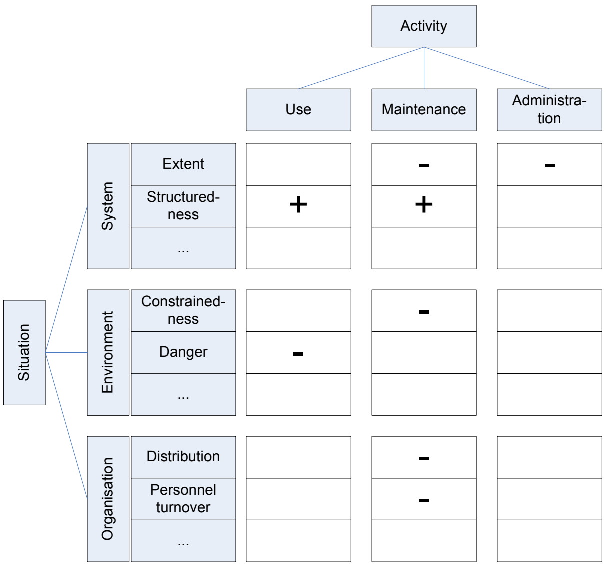
2.2 Facts and Activities

It has been proposed to use activity-based quality models (ABQM) in order to address the shortcomings of existing quality models [6]. The idea is to avoid to use high-level "-ilities" for defining quality but to break it down into detailed facts and their influence on activities performed on and with the system. In [6] this was shown for maintainability, in [17] for usability. In addition to information about the characteristics of a system, the model contains further important facts about the process, the team and the environment and their respective influence on maintenance activities such as Code Reading, Modification, or Test. For example, redundant methods in the source code, also called clones, exhibit a negative influence on modifications of the system because changes to clones have to be performed in several places in the source code.

For ABQMs, an explicit meta-model or structure model was defined in order to characterise the quality model elements and their relationships. Four elements of the meta-model are most important: Entity, ATTRIBUTE, Impact and Activity. An entity can be any thing, animate or inanimate, that can have an influence on the software's quality, e.g. the source code of a method or the involved testers. These entities are characterised by attributes such as STRUCTUREDNESS or CONFORMITY. The combination of an entity and an attribute is called a *fact*. We use the notation [Entity | ATTRIBUTE] for a fact. For the example of code clones, we can write [Method | REDUNDANCY]. These facts are assessable either by automatic measurement or by manual review. If possible, we define applicable metrics for measuring the facts inside the ABQM.

An influence of a fact is then specified by an Impact. We concentrate on the influences on activities, i.e. anything that is done with the system. For example, Maintenance or Use are high-level activities. The impact on an Activity can be positive or negative. We complete the code clone example by adding the impact on Modification: [Method | REDUNDANCY] $\xrightarrow{-}$ [Modification]. This means that if a system entity Method exhibits the attribute REDUNDANCY it will have a negative impact on the Modification activity, i.e. changing the method. A further example is the following tuple that describes consistent identifiers: [Identifier | CONSISTENCY] $\xrightarrow{+}$ [Modification] It means that identifiers that can be shown to be consistent have a positive influence on the modification activity of the maintainer of the system. In the model itself, we document more information such as textual descriptions, sources, assessment descriptions and so on. However, the short notation captures the essential relationships.

The model does not only contain the impacts of facts on activities but also the relationships among these. Facts as well as activities are organised in hierarchies. A top-level activity Activity has sub-activities such as Use, Maintenance or Administration. These examples are depicted in Figure 1. In realistic quality models, they are then further refined. For example, maintenance can have sub-activities such as Code Reading and Modification.



			Use	Maintenance	Administration
Situation	System	Extent		-	-
		Structuredness	+	+	
		...			
	Environment	Constrainedness		-	
		Danger	-		
		...			
	Organisation	Distribution		-	
		Personnel turnover		-	
		...			

Figure 1: High-level view on an activity-based quality model as a matrix

Because facts are a composite of an *entity* and an *attribute*, the organisation in a hierarchy is straightforward. Hierarchical relationships between entities do usually already exist. The top-level in Figure 1 is the situation of the software development project. It contains, for example, the System, its Environment and the development Organisation. Again, these entities need to be further refined. For example, the system consists of the source code as well as the executable. All entities can be described with attributes such as the STRUC-

TUREDNESS of the System.

The two hierarchies, the fact tree and the activity tree, together with the impacts of the facts on the activities can then be visualised using a matrix as in Figure 1. The fact tree is shown on the left, the activity tree on the top. The impacts are depicted by entries in the matrix where a "+" denotes a positive and a "-" a negative impact.

The associations between facts in the fact tree can have two different meanings. Either an entity is a part or a kind of its super-entity. Along the inheritance associations, parts and attributes are inherited. Hence, it allows a more compact description and prevents omissions in the model. For example, naming conventions usually are valid for all identifiers no matter whether they are class names, file names, or variable names.

Having defined all these entries in the ABQM, we can define which activities we want to support and which influencing facts need to be defined. In terms of the above example, if we want to support the activity Modification, we know that we need to inspect the identifiers for their consistency.

There exists a prototype tool to define this kind of large and detailed quality models [6]. Besides the easier creation and modification of the model, this has also the advantage that we can automate certain quality assurance tasks. For example, by using the tool we can automatically generate customised review guidelines for specific views.

3. ASSESSMENT AND PREDICTION APPROACH

Although activity-based quality models have proven to be useful in practice, there is no systematic approach for using such quality models in measuring. Hence, there are no quantitative assessments and predictions possible so far. We propose an approach that can be used for systematically deriving assessment and prediction models from a activity-based quality model.

3.1 Aim and Basic Idea

The general aim of the approach is to provide quality managers with a systematic method to derive assessments and predictions from an activity-based quality model. In the ABQM, there are definitions of what quality means w.r.t. different situations, artefacts, and considered activities. Currently, we give a textual description in the quality model how a fact could be assessed. For example, the fact described by [Method | REDUNDANCY] contains the following assessment description: "This fact can be assessed manually or semi-automatically. For the automatic assessments there are tools such as CloneDetective or CCFinder to detect redundant parts of the source code." This information is interesting for quality assurance planning but cannot be directly used for an overall assessment, let alone prediction.

Moreover, as the basic principle of activity-based quality models is that the most important question in quality is how well activities can be performed on and with the system, not only facts but also activities should be assessed and predicted from the knowledge of facts and impacts. Currently, only the qualitative statement whether an impact is positive or negative is made. This is suitable for rough assessments only. For more comprehensive and precise assessments of the current state and prediction of future states, a more sophisticated approach is needed. It has to systematically help to use the given relationships and enrich them with quantitative information.

As most facts and especially the relationships between facts and activities have an associated uncertainty, statistical methods are needed. Usually we cannot determine the exact relationship but can derive an uncertain range. Also values measured can be uncertain, e.g. values from expert opinion. Moreover, a statistical method is needed that can directly model the dependencies of different factors from the quality model. We identified Bayesian networks as most suitable for that task.

3.2 Bayesian Networks

Bayesian networks, also known as Bayesian belief nets or belief networks, are a modelling technique for causal relationship based on Bayesian inference. They are represented as a directed acyclic graph (DAG) with nodes for uncertain variables and edges for directed relationships between the variables. This graph models all the relationships abstractly.

For each node or variable there is a corresponding *node probability table* (NPT). These tables define the relationships and the uncertainty of these variables. The variables are usually discrete with a fixed number of states. For each state, the probability that the variable is in this state is given. If there are parent nodes, i.e. a node that influences the current node, these probabilities are defined in dependence on the states of these parents. An example is shown in Table 1. There the variable is with a probability of 60% in the state *true* if both parents are in the state *low*, and with 45% in *true* if the first parent is in *high* and the second is in *low*.

Table 1: An example NPT for a variable with two states and two parents

	low			high		
	low	med	high	low	med	high
true	0.6	0.65	0.3	0.45	0.23	0.05
false	0.4	0.35	0.7	0.55	0.77	0.95

The process of building a Bayesian network contains the identification of interesting variables that shall be modelled, representing them as nodes, constructing the topology and constructing the NPTs. Each of these steps is important and non-trivial. First, the identification of *interesting* variables includes the assumption that the model builder can decide on some basis what is interesting. In many cases, this is not clear beforehand. One possibility is to include many variables and use sensitivity analysis to remove insignificant variables.

Second, the creation of the topology uses the assumption that the model builder can decide on the dependence and independence of the identified variables. In the process of building the Bayesian network, especially for independence assumptions (i.e. missing edges in the graph) detailed justifications should be given. Third, the problem of constructing NPTs is widely acknowledged in the literature [7]. It involves defining quantitative relationships between variables. There are various possible methods for this quantification such as a probability wheel or regression from empirically collected data. All methods have their pros and cons.

It is important to note that each of these steps is important and errors in each of these steps can have a large effect on the outcome. Bayesian networks and the corresponding tool support make it easy to build models and get quantitative results. However, one needs to be aware that many assumptions are embedded in a Bayesian network that need to be validated before it can be trusted.

3.3 Four Steps for Network Building

We propose a four-step approach for building a Bayesian network as assessment and prediction model derived systematically from an activity-based quality model. The resulting Bayesian network contains three types of nodes:

- *Activity nodes* that represent activities from the quality model
- *Fact nodes* that represent facts from the quality model
- *Indicator nodes* that represent metrics for activities or facts

We need four steps to derive these nodes from the information of the ABQM. First, we identify the relevant activities with indicators based on the assessment or prediction goal. Second, influences by sub-activities and facts are identified. This step is repeated recursively for sub-activities. The resulting facts together with their impacts are modelled. Third, suitable indicators for the facts are added. Fourth, the node probability tables (NPT) are defined to reflect the quantitative relationships. Having that, the Bayesian network can be used for simulation by setting values for any of the nodes. However, we first describe the four steps in more detail.

The first step is a goal-based derivation of relevant activities and their indicators. We use GQM [2] to structure that derivation. We first define the assessment or prediction *goal*, for example, optimal maintenance planning or optimisation of the security assurance. The goal shows the relevant activities, such as *maintenance* or *attack*. This is refined by stating *questions* that need to be answered to reach that goal. For example, will the maintenance effort increase over the next year or how often will there be a successful attack in the next year? Finally, we derive *metrics* or indicators that allow a measurement to answer the question. In the examples, that can be effort change trend or number of harmful attacks over the next year.

In the second step, we use the quality model to identify the other factors that are related to the identified activities. There are two possibilities: (1) there are sub-activities of the identified activities and (2) there are impacts from facts to the identified activities. We repeat this recursively for the sub-activities until all facts that have an impact on the activities sub-tree below the identified activity are collected. For each activity we immediately see the impacts and hence the corresponding facts. All activities and facts identified this way are modelled as nodes in the Bayesian network. We add edges from sub-activities to super-activities and from facts to activities on which they have an impact. Figure 2 gives an abstract overview of that mapping from the quality model to the Bayesian network.

In the third step, we add additional nodes as indicators for each fact and activity node that we want a measurement for. The indicator for our relevant activity was defined in the first step. Hence, we can add additional indicators for sub-activities if needed. In any case, there need to be at least one indicator for each fact that is modelled. There might be a precise description in the quality model already. Otherwise, we need to derive our own metric or use an existing one from the literature. The edges are directed from the activity and fact nodes to the indicators, i.e. the indicators are dependent on the facts and activities. An indicator is only an expression of the underlying factor it describes.

A main advantage of using a ABQMs as a basis for the Bayesian network is that it prescribes its topology. One of the main points of such quality models is to qualitatively describe the relationships between different factors that are relevant for software quality. We rely on that and assume that all dependencies have been modelled and that all other factors are independent. On the one hand, this constrains the validity of the results of the Bayesian network by the validity of the ABQM. On the other hand, it frees the network builder from reasoning about independence and dependence.

Finally, the fourth step, enriches the Bayesian network with quantitative information. This includes defining node states as well as filling the NPT for each node. The activity and fact nodes are usually modelled as *ranked* nodes, i.e. in a ordinal scale. The most common example is low, medium, and high. This has advantages in evaluation and aggregation. The evaluation is easier as not precise numbers have to be determined but coarse-grained levels. These levels actually reflect much more the high uncertainty in the data. Also in aggregating over nodes (up the hierarchy in the activity tree) coarse-grained ranked data is more simple to handle. For only a few levels in ordinal scale, it is easier to define a aggregation specification than for continuous data. To define the NPT, we use an approach proposed by Fenton, Neil and Galan Caballero [9]. The basic idea is to formalise the behaviour observed with experts that have to estimate NPTs. They usually estimate the central tendency or some extreme values based on the influencing nodes. The remaining cells of the table are then filled accordingly. This is similar to linear regression where a Normal distribution is used to model the uncertainty. We use the doubly truncated Normal distribution that is only defined in the $[0, 1]$ region. It allows to model a variety of shapes depending on the mean and variation. It allows, for example, to model the NPT of a node by a weighted mean over the influencing nodes.

The node states of indicator nodes depend on the scale of the indicator used. This often will be continuous or discrete interval states such as lines of codes in intervals of a hun-

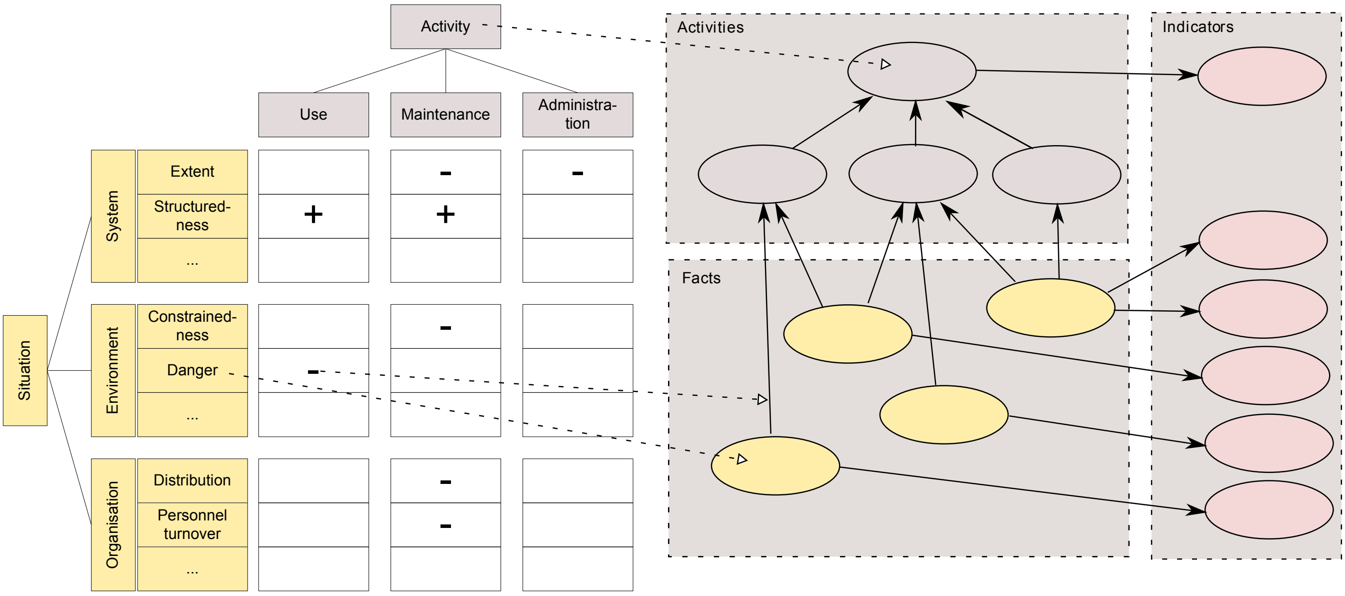


Figure 2: Mapping of quality model elements to Bayesian network elements. The dashed arrows indicate examples for mappings.

dred. The NPTs of the indicator nodes are then defined using either common industry distributions or information from company-internal measurements. For example, typical LOC distributions can be accumulated over time. The influence of the activity or fact node, respectively, it belongs to can be modelled in at least two ways: (1) partitioned expressions and (2) arithmetic expressions. The latter describes a direct arithmetical relationship from the level in the activity or fact node to the indicator. Using a partitioned expression, the additional uncertainty can be expressed by defining probability distributions for each level of the activity or fact node.

3.4 Usage of the Bayesian Network

A main feature of Bayesian networks is their capability to simulate different scenarios. Having built the Bayesian network based on the activity-based quality model, we can ask "what if?" questions. These questions are formulated in scenarios that can be simulated and compared. A scenario involves adding additional information to the model, more specifically, adding an observation for a node. This way, the uncertainty is removed and the consequences for the other nodes can be calculated. Interestingly, in a Bayesian network it is possible to do forward as well as backward inference, i.e. information can be added to any node and the effect is calculated in any direction of the graph.

A first straightforward scenario is to add the currently measured values for the fact indicator nodes. This will drive the calculation up to the activities and the activity indicator node. This indicator node then shows the probability distribution for its value, i.e. the value of the activity. Then changes can be made to the fact indicator nodes in other scenarios to reflect possible changes. Their effect on the activity can be predicted. Another interesting scenario is to set a desired value for the activity indicator and let the network calculate the most probable explanation in the fact indicators. It shows the values that should be reached in order to fulfil the goal.

4. PROOF OF CONCEPT

The above presented approach can be used in many different contexts and to answer various assessment and prediction questions. We provide a proof of concept here that applies the approach to a publicly available data set.

4.1 Goal

Currently, we cannot provide a full validation of the assessment and prediction approach because this would involve measuring a large number of facts from the quality model in order to take full advantage from the knowledge contained in it. This data is not available in public data sets. Measuring this at a company will need time and effort. Only then a sensible analysis of the predictive validity and comparisons with other prediction models are possible.

Nevertheless, we provide a proof of concept application of the approach based on a small extract of our quality model for maintainability for which there is publicly available data. It demonstrates the basic principles of the approach on a real data set. This way, we can analyse whether the approach is feasible in an almost realistic setting. The predictive validity in the proof of concept can give an indication of the usefulness of the approach. There should at least be an improvement over industry average values.

4.2 Context

The proof of concept comprises the prediction of the maintainability of a system. The system under analysis was developed by NASA in the project CM1 for which the data has been publicised [12]. The system itself is an instrument in a space craft developed in C. Various metrics, especially the McCabe and Halstead metrics, have been collected in this project.

As the quality model, we use the activity-based quality model for maintainability from [6]. It contains a complete activity tree for maintenance as well as about 200 facts with an impact on those activities. The CM1 data set does not contain data for all of these facts but we choose a small set of facts and corresponding activities to predict maintainability. The choice was therefore guided by the availability of the data. The actual maintainability can be judged for CM1 because the effort for several changes has been documented. We will use this as our surrogate measure.

For modelling the Bayesian network, we use the tool AgenaRisk[1]. It provides a complete tool environment for Bayesian networks including the usage of expressions for describing the NPT and sensitivity analysis.

4.3 Procedure

The first step in our assessment and prediction approach is to identify the relevant activities and corresponding indicators (metrics) in a GQM-like procedure. We want to analyse maintainability and we assume that the quality manager is interested in the question of how high the maintenance efforts will be in the future. This information helps in planning the maintenance team. Hence, we formulate the **Goal** as "Planning of future maintenance efforts". We can directly identify the activity Maintenance in that goal. To further operationalise that, we define the corresponding **question** as "What will be the maintenance effort per change request?". This information, probably together with a prediction of the yearly number of change requests, would give a good basis for maintenance planning. However, we concentrate only on the question about the effort per change request. This leads us straightforwardly to the **metric** "average effort per change request". Hence, an analysis of the mean is needed.

In the second step, we already start building the Bayesian network. We look at the maintainability model and find no impacts directly on Maintenance that should be considered here. However, we find 10 sub-activities including the following 3 that we will further analyse: Quality Assurance, Implementation and Analysis. All three have impacts from facts but having in mind the available data, we ignore these and use the further sub-activities Testing, Modification and Comprehension with its child Code Reading. We create nodes for these activities and connect them with edges corresponding to their hierarchy. They can be seen in Figure 3 in the box "Activities".

We only include 3 impacts on each of the lowest level activities chosen so far because we can find corresponding data in the CM1 data set. These facts together with their impacts are:

- [Module | EXTENT] $\xrightarrow{-}$ [Code Reading]
  The size of a module has an impact on reading the code of the module. In essence, the larger the module, the longer it takes to read it.
- [Implementation | REGULARITY] $\xrightarrow{+}$ [Testing]
  An implementation is regular if it does not use unnecessarily nested branches. This complex structure would render coverage by tests more difficult.
- [Comment | APPROPRIATENESS] $\xrightarrow{+}$ [Modification]
  Comments need to appropriately describe the code it is associated with.

The facts are added as nodes in the Bayesian network. In Figure 3 they are contained in the box "Facts". The impacts are included as the arrows from the facts to the activities.

The indicators that are identified in the fourth step of the approach are taken from the available data of the data set. We only use 1 indicator per fact although we are aware that each fact has more aspects that should be covered. For the *Extent of Modules*, we use the indicator *Average module size* given in LOC. The *Regularity of the Implementation* is indicated by the *Average Cyclomatic Complexity*. This is not a particularly good indicator as it only gives a number for the decision points in the implementation. A manual review can far better decide whether the implementation is regular. However, we have no access to review results. A similar reasoning holds for the indicator *Comment Ratio* for the fact *Appropriateness of Comments*. The proportion of the comments in relation to the other code is only of minor importance in comparison with the semantic appropriateness. However, we do not have access to such a semantic judgement. The indicators can be found in Figure 3 on the right-hand side in the box "Indicators".

A difficult problem in general with Bayesian networks is the definition of the node probability tables (NPT) [7]. Various method can be used to define these NPTs that we will not describe here in detail. For the approach, we can simplify the problem to 2 cases: (1) activities and facts NPTs and (2) indicator NPTs. The NPTs for the activities and facts can, unless there is additional knowledge, be assumed as uniformly distributed. The impacts are only modelled as direct influence. Here, a special method for ranked nodes [9] can be applied that simplifies the task. For the indicator NPTs either empirically investigated distributions of the company or industry average distribution should be used. This forms the distribution of the indicator values under uncertainty without any observations.

For the average effort of a change, we refer to [15] that gave an a mean defect removal cost of 27.4 person-hours with minimum 3.9 and maximum 66.6. Although a change does not always have to be a defect removal, it is precise enough for the proof of concept. For the distributions of the other indicators, no published distributions were available. Hence, our own expert opinion was used as a basis.

4.4 Results and Discussion

Figure 3 shows two scenarios. The green values describe the scenario in which no observations have been made, i.e. the general scenario. The blue values are for the scenario *Measured values* in which for the three fact indicators the real measured values are set as observations. More specifically, the comment ratio is set to .2517, the average cyclomatic complexity to 5.18 and the average module size to 33.47 LOC. This has effects on the rating of the facts in turn on the activities. The most interesting value, however, is the

[1] http://www.agenarisk.com/

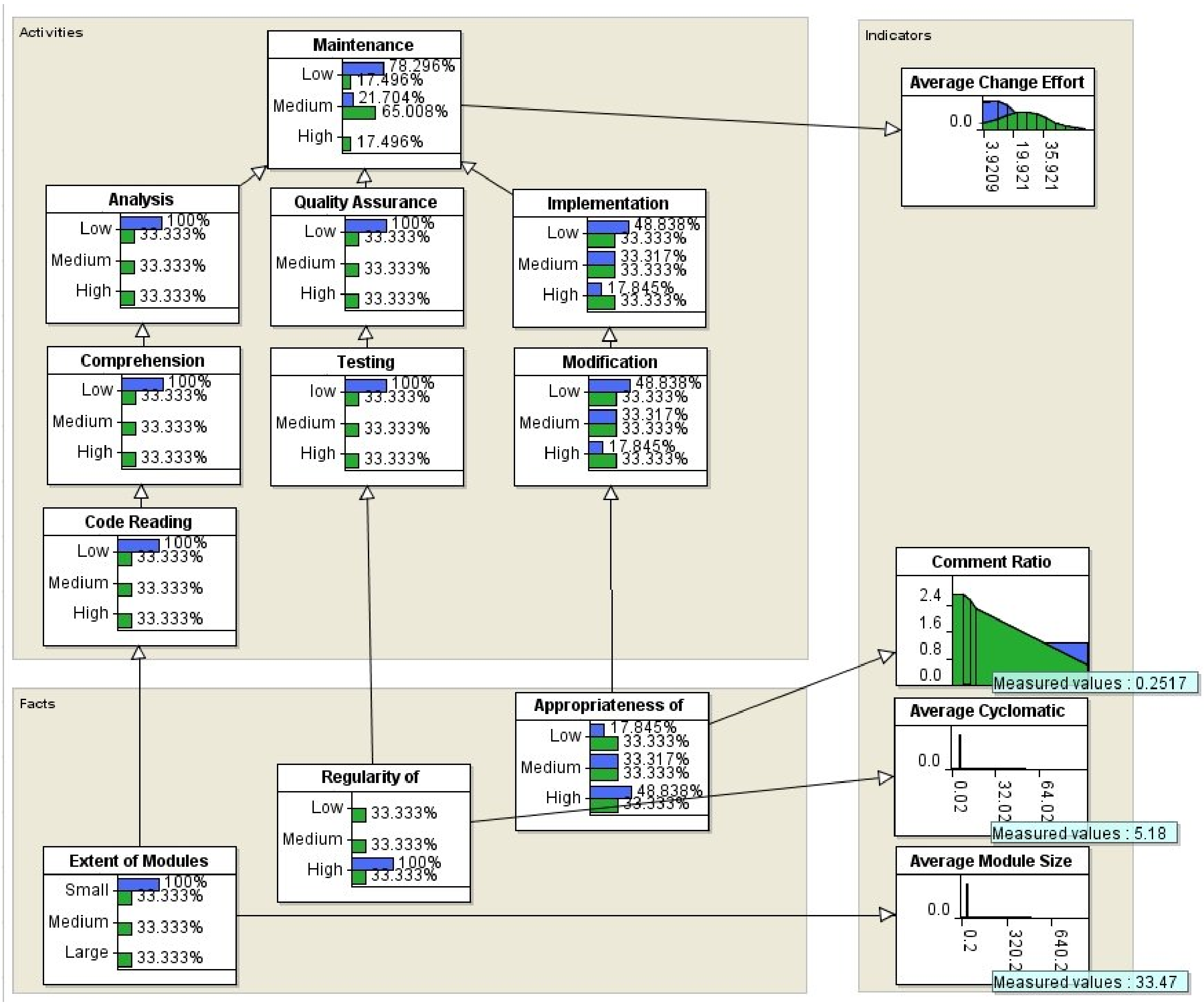


Figure 3: The resulting Bayesian network

average change effort. In the general scenario, this variable has a mean value of 27 with a standard deviation of 12.1. The additional information about the measured values shifts the distribution to the left. The mean decreases to 15.9 with standard deviation 8.5. This is still not the really measured value of 6 but much closer than the industry standard of 27. Hence, it is an improvement in comparison to just using standard numbers.

The reason for the large difference between the prediction and the real value is probably three-fold. First, the distribution from [15] might not be appropriate for the NASA environment in which the effort for a change request seems significantly smaller than average. Second, it might also be the case that the data in the CM1 data set does not use exactly the same measures as in [15]. The degree to which additional efforts for re-inspection and re-testing are included could vary. Third, several more factors than the 3 considered might have an influence on the maintenance effort. The quality model contains many explanations in terms of facts that should be investigated.

Nevertheless, the point of the proof of concept was not mainly to show predictive validity but to investigate the general feasibility. We saw that it is possible with reasonable effort to build a Bayesian model. The 4-step approach gives direct guidance for most of the network building. Only setting up the NPTs is still a challenge. There are usually several possibilities how a relationship can be expressed and with how much uncertainty it should be afflicted. This still needs expert opinion and experimentation. Nevertheless, AgenaRisk provides very good tool support to find easier ways to define an NPT. As our ABQMs can get very large with a few hundreds of model elements, it remains to be evaluated whether the approach scales when the quality model is fully mapped to a Bayesian network. Probably, a selection of a subset of the quality model is necessary first.

Furthermore, it is important to note that a more in-depth validation of a resulting Bayesian network is necessary in order to ensure that all parts, topology, node states, and NPTs, represent the interdependencies of the quality factors good enough so that a valid statement about the quality of the software system can be made. This is not covered by this proof-of-concept but has to be the next step.

5. RELATED WORK

The basic idea to use Bayesian networks for assessing and predicting software quality has been developed mainly by Fenton, Neil and Littlewood. They introduced Bayesian networks as a useful tool and applied it in various contexts related to software quality. In [8] they formulate a critique on current defect prediction models and suggest to use Bayesian networks. Other researchers also used Bayesian networks for software quality prediction similarly [1, 14]

The work closest to the approach proposed in this paper is [13]. They discuss quality models such as the ISO 9126 [10] and their problems such as the undefinedness of the relationships in such a model. They aim at solving these problems by defining Bayesian networks for quality attributes directly. Our work differentiates in using a defined structure for quality models that contain far more details as common quality models. This structure and detail allows a straightforward derivation of a Bayesian network from the quality model. This has the advantage that the basic quality model can also be used for other purposes then prediction such as the specification of quality requirements.

Beaver, Schiavone and Berrios [4] also used a Bayesian network to predict software quality including diverse factors such as team skill and process maturity. In his thesis [3], Beaver even compared the approach to neural networks and Least Squares regressions that both were outperformed by the Bayesian network. However, they did not rely on a structured quality model as in our approach.

6. CONCLUSIONS

A high goal in software quality management is the reliable quantitative assessment and prediction of software quality. Many efforts in building assessment and prediction models have given various insights in the usefulness but also the constraints of such models. However, these models have not been tightly integrated into other quality management activities. Activity-based quality models have proven in practice to be a solid foundation for defining quality on a detailed level. However, quantitative analyses have not been directly possible so far.

Bayesian networks have been shown to provide promising results in quality predictions. Because of that and their clear structuring that can straightforwardly reflect the structure of activity-based quality models, a 4-step approach for transferring activity-based quality models to Bayesian networks was proposed. It allows to systematically construct a Bayesian network that uses the knowledge encoded in the quality model to provide information about a given assessment or prediction goal. In the terminology of [5], we use a *quality definition model* (the ABQM) and enrich it with a *quality assessment* and *quality prediction model* (the BN).

Although not fully validated, we demonstrated the approach in a case study using real NASA project data. The proof of concept showed the applicability of the approach on such a project and even could improve the prediction in comparison to industry standard values. The prediction was not very accurate but that can have various reasons such as a different effort distribution at NASA or the influence of more factors that have not been considered because there was no data available.

The use of Bayesian networks opens many possibilities. Most interestingly, after building a large Bayesian network, a sensitivity analysis of that network can be performed. This can answer the practically very relevant question which of the factors are the most important ones. It would allow to reduce the measurement efforts significantly by concentrating on these most influential facts.

We plan to apply this approach in future case studies at our industrial partners in order to further validate the approach. A comprehensive analysis of the predictive validity is necessary to judge the usefulness of the approach and to compare it with other means for assessment and prediction.

7. ACKNOWLEDGMENTS

This work has partially been supported by the German Federal Ministry of Education and Research (BMBF) in the project QuaMoCo (01 IS 08023B).

8. REFERENCES

[1] S. Amasaki, Y. Takagi, O. Mizuno, and T. Kikuno. Constructing a Bayesian Belief Network to Predict Final Quality in Embedded System Development. *IEICE Transactions on Information and Systems*, E88-D(6):1134–1141, 2005.

[2] V. R. Basili, G. Caldiera, and H. D. Rombach. Goal question metric paradigm. In J. C. Marciniak, editor, *Encyclopedia of Software Engineering*, volume 1. John Wiley & Sons, 1994.

[3] J. M. Beaver. *A life cycle software quality model using bayesian belief networks*. PhD thesis, University of Central Florida, 2006.

[4] J. M. Beaver, G. A. Schiavone, and J. S. Berrios. Predicting software suitability using a bayesian belief network. In *Proc. Fourth International Conference on Machine Learning and Applications*, pages 82–88. IEEE Computer Society, 2005.

[5] F. Deissenboeck, E. Juergens, K. Lochmann, and S. Wagner. Software quality models: Purposes, usage scenarios and requirements. In *Proc. 7th International Workshop on Software Quality (WoSQ '09)*. IEEE Computer Society, 2009.

[6] F. Deissenboeck, S. Wagner, M. Pizka, S. Teuchert, and J.-F. Girard. An activity-based quality model for maintainability. In *Proc. 23rd International Conference on Software Maintenance (ICSM '07)*. IEEE Computer Society Press, 2007.

[7] N. Fenton and M. Neil. Managing risk in the modern world. Applications of Bayesian networks. Knowledge transfer report, London Mathematical Society, 2007.

[8] N. E. Fenton and M. Neil. A critique of software defect prediction models. *IEEE Transactions on*

Software Engineering, 25(5):675–689, 1999.

[9] N. E. Fenton, M. Neil, and J. G. Caballero. Using ranked nodes to model qualitative judgments in Bayesian networks. *IEEE Transactions on Knowledge and Data Engineering*, 19(10):1420–1432, 2007.

[10] ISO 9126-1. Software engineering – Product quality – Part 1: Quality model, 2001.

[11] B. Kitchenham and S. L. Pfleeger. Software quality: The elusive target. *IEEE Software*, 13(1):12–21, 1996.

[12] NASA IV&V Facility. Metrics data program. `http://mdp.ivv.nasa.gov/`.

[13] M. Neil, B. Littlewood, and N. Fenton. Applying bayesian belief networks to systems dependability assessment. *Safety Critical Systems Club Newsletter*, 8(3), 1999.

[14] E. Perez-Minana and J. Gras. Improving fault prediction using Bayesian networks for the development of embedded software applications. *Software Testing Verification and Reliability*, 16(3):157, 2006.

[15] S. Wagner. A literature survey of the quality economics of defect-detection techniques. In *Proc. 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE '06)*. ACM Press, 2006.

[16] S. Wagner and F. Deissenboeck. An integrated approach to quality modelling. In *Proc. 5th International Workshop on Software Quality (5-WoSQ)*. IEEE Computer Society Press, 2007.

[17] S. Winter, S. Wagner, and F. Deissenboeck. A comprehensive model of usability. In *Proc. Engineering Interactive Systems 2007*, volume 4940 of *LNCS*, pages 106–122. Springer-Verlag, 2008.