Software Quality Modeling using Metrics of Early Artifacts

Chandan Kumar1, Dilip Kumar Yadav2

1,2Department of Computer Applications National Institute of Technology, Jamshedpur, India-831014

1chandan.jha150286@gmail.com, 2dkyadav1@gmail.com

ABSTRACT
Software industries require reliability prediction for quality evaluation and resource planning. In early phase of software development, failure data is not accessible to conclude the reliability of software. However, early software fault prediction procedure provides a flexibility to predict the faults in early stage. In this paper, a software faults prediction model is proposed using BBN that focus on the structure of the software development process explicitly representing complex relationship of five influencing parameters (Techno-complexity, Practitioner Level, Creation Effort, Review Effort, and Urgency). In order to assess the constructed model, an empirical experiment has been performed, based on the data collected from software development projects used by an organization. The predicted fault ware found very near to the actual fault detected during testing. Keywords: Software Defect Prediction, Software Reliability, Bayesian Belief Network (BBN)

1. INTRODUCTION
A software artifact can be released only after some pre-specified reliability criterion has been met at the end of the development cycle. A number of logical models have been projected for assessing the reliability of a software system [1]. Data-driven approach and logical method are used to develop software reliability model. Normally, a good number of models consider fault detection process and data for analysis. In the early phase before testing, it is difficult to obtain accurate predictions. But early software defect prediction is useful for assessment of the software development process. In early software reliability prediction literature, there are numerous approaches and models studied like Causal Model (Bayesian net) [11, 12], ProPRED (Probabilistic Model) [13], ESRA (Early Software Reliability Assessment) approach [2, 3, 4], UML based software models [5], ERAT (Early Reliability Analysis Technique) [6], ERPM (Early Reliability Prediction Model) [7], Enhanced Model [8], Decision Tree Based Model in combination of K-means clustering as preprocessing technique [9]. There are no methods and approaches universal for all possible software projects. To develop a reliable software system, several issues are to be addressed. These include depth of reliable software, regular development methodologies, testing methods for reliability, reliability growth prediction modeling, and accurate assessment of reliability. The issue of finding a universal model for all possible software projects is yet to be solved. Selection of an exact model is very important in software reliability prediction because both the release date and the resource allocation assessment can be affected by the accuracy of prediction. The BBN is one of the methods for modeling systems that include underlying relationships among variables. The benefit of a BBN model is that it is not compulsory to enter values for all of the factors in order to get a prediction. In fact, for this model, reasonably exact predictions of the faults can be achieved typically by entering values for just the three or four most important factors recognized in the sensitivity analysis. Hence, this category of model can be used for competent decision-making and trade-off analysis during early development phases. In general BBN modeling, a Bayesian belief value to each unsure event is assigned, such as the belief value of “Cloudy” is assigned as 0.65, the belief value of “Jony is late” is 0.2, and so on. All these probabilities for the unsure events come from people’s personal judgments or domain experts that are determined by collecting empirical, historical or statistical data. The rest of the paper is ordered as follows: in section II, we present a concise analysis on Bayesian Belief Networks. In section III, we proposed a BBN model for the software development process, and then we evaluate the model by applying the authentic project data in section IV. Experiments and results are described in section V and conclusion is given in section VI.

2. BAYESIAN BELIEF NETWORK (BBN)
A Bayesian Belief Network (BBN) [10] represents underlying relationships of a system or dataset and provides a graphical representation of this causal structure through the use of Directed Acyclic Graphs (DAGs) with nodes and edges. The DAG representation then provides a framework for assumption and prediction. The nodes represent random variables with probability distributions, while edges represent weighted underlying relationships between the nodes. Each node has a probability of having a certain value. A directed edge exists from a parent to a child. Each child node has a conditional probability table based in parental values. An example of DAG is shown in Fig. 1. This represents an underlying relationship between
3 random variables P, Q and R whereby there is a relationship P→Q and P→R but there is no direct relationship between Q and R. The existence of the underlying relationship P→Q is represented by the arc/edge between the nodes P and Q while the strength of the relationship is represented by the conditional probability P (Q / P).

Fig. 1. BBN underlying structure

The probability table is assigned to each variable. The probability table is especially assigned to the dependent variable (here, Q, R and T are such dependent variables), and is named the conditional probability table (CPT). It is assumed that all variables are binary which take values “True” or “False” and values of some variables are unknown Fig. 2. The BBN approach provides quantified and auditable risk estimation and enables combination of multiple forms of data. This includes inflexible data such as test results, as well as individual data such as experience of staff. The most important advantages accessible by BBN are:

1) Best method for analysis under uncertainty.
2) Prediction based on evidence propagation algorithm that dynamically updates the model given all current data.
3) Can come together different data, including individual beliefs and empirical data.
4) Will provide predictions even if there is partial or missing data.
5) Visual analysis that makes every assumptions and evidence explicit and auditable to regulators and inspectors.
6) Achieve powerful "what-if" analysis to test sensitivity of conclusions.
7) Be capable of used now on real, large-scale problems since computational tractability issues have been solved.

Fig. 2. Collecting facts in BBN

3. BBN MODEL OF SOFTWARE DEVELOPMENT PROCESS

Defining the BBN Model: To define the BBN model, there is a need to identify the range of information and data sources available, and identify risky variables. The five metrics considered in this model are described below.

3.1. Techno-complexity
This metric describes the technology used for the software development and the complexity of software requirements. The increased complexity lead to misinterpretation about the product features.

3.2. Practitioner level: (Experience + Product familiarity)
This metric describes the experience of the software development team members and their familiarity with the software product to be developed. The programmers having less experience and less personal knowledge or information about the product features are more prone to errors and omissions.

3.3. Creation Effort
Creation effort is assessed in terms of the person-hours. The increased effort spent in the creation of the software requirement specification document, design phase and in the coding phase introduces a smaller amount of faults in the software.

3.4. Review effort
Review effort is assessed in terms of the person-hours. The better effort spent in the review of the software requirement
specification document, design phase and in the coding phase extracts out the faults in the software.

3.5. Urgency: Percentage compression in time.
Assuming that more number of faults are encountered if the time span for completion of the software is less. The architecture of the proposed model has been shown in Fig. 3.

4. MODEL EVALUATION
To evaluate our model we enter conditional probabilities to the various nodes of the BBN which are obtained with the help of experts working in the specified domain. The complied mode of BBN is shown in the Fig 4.

5. EXPERIMENTS AND RESULTS
Model is experimented with three different specifications of projects and gets the results with the help of Netica software [14]. The specifications of projects are shown in Table 1-3.

<table>
<thead>
<tr>
<th>Table 1. Specifications for Project 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sr. No.</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>Actual Defects</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2. Specifications for Project 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sr. No.</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>
Table 3. Specifications for Project 3

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Influencing Variables</th>
<th>SRD</th>
<th>Design</th>
<th>Coding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Quality of artifacts made+ Clarity of requirements</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>2</td>
<td>Effort spent for review w.r.t creation effort</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>3</td>
<td>Effort spent for creation w.r.t ideal phase effort</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>4</td>
<td>Experience of practitioner of the relevant stage</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>5</td>
<td>Technology/ Novelty of the feature</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>6</td>
<td>Familiarity of practitioner with product family</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>7</td>
<td>Urgency/standard time</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>8</td>
<td>Complexity of application</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>9</td>
<td>Unplanned interruptions during the work</td>
<td>NA</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>10</td>
<td>Detected Faults</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>Actual Defects</td>
<td></td>
<td></td>
<td>18</td>
</tr>
</tbody>
</table>

The results above show better fault predicting accuracy of BBN model with around 81% correlation with the actual faults.

6. CONCLUSION

The proposed model is developed using the qualitative value of five metrics (Techno-complexity, Practitioner Level, Creation Effort, Review Effort, and Urgency) of early artifacts of software development life cycle. Predicted number of defects is very near to actual defects detected in testing phase for three software projects shown in Table 2. The model is helpful during testing phase for planning the testing resources.

7. REFERENCES


