Prediction of Software defects in SDLC using BN

Jawahar Sambhaji Gawade
Information Technology Dept.,
SVPMs College Of Engg.,
Malegaon(Bk), Baramati, Pune

Dinesh Bhagwan Hanchate
Computer Engineering Dept.,
Vidya Pratishthans College Of Engg.
Baramati,Pune, India.

ABSTRACT
This project reviews the use of Bays Networks (BNs) in software defects Prediction. The idea allows us to incorporate causal process factors. It does the combination of qualitative and quantitative software measures. It stops to play some well-known traditional software metrics methods limitations. Decision support tools for this have been built using causal models represented by Bays Networks (BNs), incorporate empirical data and judgment of experts. Previously, this required a custom BN for each development lifecycle phase. We described a more general idea that allows causal models to be applied to any lifecycle phases. The approach is evolved through collaborative projects and captures significant commercial input. For software projects within the range of the models, accuracy of defect predictions are very good. The main functions provided to the end-user is observations and can be entered using a questionnaire interface, where questions are concerned to Bays Network variables. The model predicts the defects likely to be left in software after testing. The model uses the results of statistical analysis on the Previous software projects. It can be combined with other defect prediction models to predict the number of residual defects of different categories. The Bayesian network structure is, here, a set of project domain conditional independence relation. BN learning structure which represents a domain. This domain can light on its underlying causal structure. This results in significantly improved accuracy for defects and reliability prediction type models.

General Terms:
Software Testing, Machine learning

Keywords:
Software defects, BN (Bayesian network), Defect Prediction

1. INTRODUCTION

\texttt{L} A software defect is due to the result of an error or bugs. It is an disorder cause which may lead to abnormal software behavior not according to its specification. Causal models are very important because they allow all the evidence to be taken into account, even when Confliction of different evidence \([11][12]\). It is assumed that few defects are found during testing, does this mean that testing is poor or that development was outstanding and the software has few defects to find. The Regression-based models of software defects are little helpful to a Project Manager and Project Members who must decide between these alternatives \([11]\). Previous projects database is used to build the BN, with judgment of Experts on the strength of each causal mechanism. In this paper, We worked by describing a much more flexible and general method of using BNs for defect prediction \([11][7]\).

2. METHODOLOGY \([9][11]\)

2.1 A. Baysian network \([3]\):
Bays network is a graph together with an associated set of probability tables of Class. The BN nodes represent uncertain variables. The relevance relationships between the variables are represented by arcs. The Figure 1 shows the Bayesian Network forms a causal model of the process of inserting, finding and fixing software defects. The variable Probability of effective KLOC implemented represents the complexity adjusted size of the functionality implemented \([13]\). The amount of Functionality increases the number of potential defects also rises. The probability of avoiding defect in development determines defects in development phase. This number represents the number of defects before testing which are in the reviewed and revisited code that has been implemented. The inserted defects may be found and fixed. The other residual defects are those remaining after testing phase. Variables representing a number of defects take a value in a numeric range, discarded into numeric interval. There is a probability table for each node, specifying how the probability of each state of the variable depends on the states of its BN class. Some of these are deterministic: for ex-
The main challenging job of decision during SDLC are
• to apply finite resources to all of these activities.
• to confirm the dependency depending upon on the classification and applied resource division.
• to Predict the likely quality that will be achieved when the product is delivered.

Software project manager, quality manager has to make the correct decision mostly as nearly all software projects rely upon the judgment of the project or quality manager. The proposed work is presented in a single model to combine diversified evidences (called causal) in SDLC in more user free and efficient way. Graphical probability models are used (also known as Bayesian Belief Networks) as the appropriate formalism for representing this evidence [10] [13]. It is also possible to use the corresponding decisions and judgments of experienced Quality and Project Managers to have probabilistic model. This model can be utilized to put up some conclusions about Software Quality throughout the SLDC. The BN represents the complete joint probability distribution assigning a probability to each combination of states of all the variables but in a factored form, greatly reducing the space needed. When the states of some variables are known, the joint probability distribution can be recalculated conditioned on this evidence and the updated marginal probability distribution over the states of each variable can be observed. The quality of the development and testing processes is represented in the BN of Figure 1 by four variables discredited over the 0 to 1 interval [10] [13] [6]. These are [4]
• probability of avoiding specification defects,
• probability of avoiding defects in development,
• probability of finding defects,
• Probability of fixing defects.

The BN in Figure 1 is a simplified version of the BN at the heart of the decision support system for software defects. None of these probability variables are entered directly by the user instead these variables have further parents modeling the causes of process quality.

2.2 B. SDLC and phases [12] :
A development lifecycle is model, and is made up from phases, but a phase is not a fixed development process as in the traditional waterfall lifecycle, a phase can consist of any number and combination of such development processes. Each phase then includes all the development processes: specification, design, coding and testing. Even in a traditional waterfall lifecycle it is likely that a phase includes more than one process with, for example, the
testing phases involving some new design and coding work [2].

To cover all parts of this continuum, we considered all phases to include one or more of the following development activities [2] [11]:

- **Specification/documentation**: This covers any activity whose objective is to understand or describe some existing or proposed functionality. It includes: requirements gathering, writing, reviewing, or changing any documentation (other than comments in code).
- **Development (or more simply coding)**: This covers any activity that starts with some predefined requirements (however vague) and ends with executable code.
- **Testing and rework [5]**: This covers any activity that involves executing code in such a way that defects are found and noted; it also includes fixing known defects.

Figure 2 shows the system architecture of a project in which project manager or team members enter detailed information about every project. Each project has five BN classes described below which have set of various questionnaires with scale. Bayesian Networks created by using Combination of Phase wise Questionnaire. Residual defect post contains output of combination of BN classes. Finally, the graph representation shows the comparison of projects: phase based, project based and domain based. The phase BN includes all these activities, allowing the extent of each activity in any actual phase to be adjusted. In the most general case, a software project consists of a combination of these phases.

### 3. EXPERIMENT AND RESULT

The phase BN is constructed from five classes [1][11][12]:

- One of three activity classes: specification and documentation, design and development, and test and rework.
- The scale of New functionality developed in this phase.
- The defect insertion and discovery.

Fig. 3 shows a single object instantiation of each of these classes. This object view of the single phase model represents the BN in abstract terms. The inner details of each class are not shown only the input and output nodes are available (The inner details are available with authors). In this view, a class is represented by its interface to other classes [11] [7]. Triangular arrow heads represent input nodes within a class, whereas rounded arrow tails represent output nodes. Lines represent input node instantiation, i.e. the output node of one object instantiates (replaces) the input node of the connected object. Input nodes effectively act as parameters for a BN class. Note that not all input nodes are instantiated by output nodes from another object. Input nodes have a default probability distribution associated with them [13]. However, this is rarely used. More often, unattached input nodes are initialized using explicit observations. E.g., Residual defects pre is used to account for defects remaining from previous phases. If this is the first or only phase, then it should be explicitly initialized to zero.

#### 3.1 BN CLASSES [13] [7] [6]

Following BN Classes are developed for the proposed approach [14].

- **New Functionality is Implemented**: Since we are to build and test some software we may be implementing some new functionality in this phase. This class provides a measure of the size of this functionality.
Fig. 4. Input: Project information

Fig. 5. Input: Specification Phase Questionnaires
Figure 6. Input: Bayesian Network for Specification

• Specification and Documentation. This class is concerned with measuring the amount of specification and documentation work in the phase, the quality of the specification process and determining the change in the quality of the documentation as a result of the work done in the phase.
• Design and Development. This class models the quality of the design and development process, which influences the probability of inserting each of the potential defects into the software.
• Testing and Rework. This class models the quality of the testing process and the rework process, influencing the probabilities of finding and fixing defects.
• Defect Insertion and Discovery.

3.2 Input snapshots

Figures 3 to 6 represents input to the model designed.

3.3 Output snapshots

These are shown in Figures 7 to 10. Result Screenshots (a) Bayesian Network for Residual Defects Post (b) Project based Comparison (c) Domain based Comparison (d) Phase based Comparison.

4. CONCLUSION

We have shown how a wide variety of software lifecycles can be modeled using a DBN in which each time frame is a lifecycle phase combining all software development activities in different amounts. This idea allows a BN for software defect prediction to be tailored to different software development environments. From the industrial point of view, there is a need to generate predictive models fast. Thus, any kind of automation of this process is desired. Bayesian nets can be built in various ways: by an expert, purely from the data, or by combining expert knowledge with empirical data. This Project approach allows a BN for software defect prediction to be tailored to different software development environments. This will be very useful to predict and avoid the defects in different lifecycle
phases of software development. In the absence of an extensive and expensive reliability testing phase, this model can be used to provide an estimate of residual defects that is sufficiently precise for many software project decisions.

5. REFERENCES


