International Journal of Computer Applications (0975 8887)
Volume 99 - No. 4, August 2014

Prediction of Software defects in SDLC using BN

Jawahar Sambhaji Gawade
Information Technology Dept.,
SVPMś 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

LaTeX 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] [8]. 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].
· In section II, we described Methodology to be used along with BN and SDLC.
· In section III, Experiment,result is illustrated with input and output snapshots of our approach.
· Section IV concludes the paper.
We, also, described how the Questionnaire set is used to create an effective decision support system from the BN. An important limitation of the earlier work was the need to build a different BN for each software development lifecycle to reflect both the differing number of testing stages in the lifecycle and the differing metrics data available. To overcome this limitation of earlier work, we described a BN that models the creation and detection of software defects without commitment to a particular development lifecycle. We, then, showed how a software development organization can adapt this BN to their development lifecycle and metrics data with much less effort than is needed to build a tailored BN from scratch [11] .

2. METHODOLOGY [9] [1]

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, discredited 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-

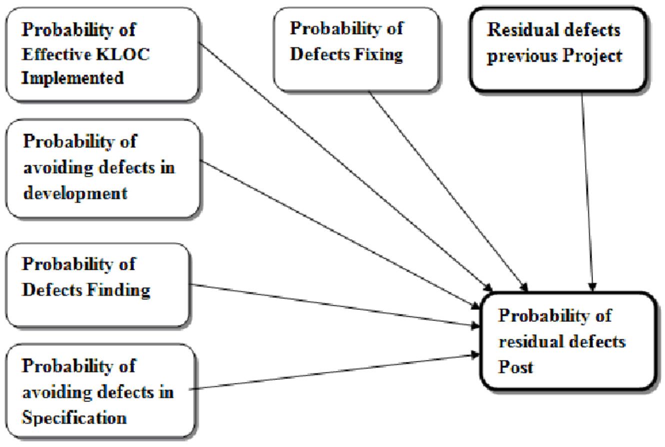

Probability of Effective KLOC Implemented
Probability of Defects Fixing
Residual defects previous Project
Probability of avoiding defects in development
Probability of Defects Finding
Probability of residual defects Post
Probability of avoiding defects in Specification

Fig. 1. BN for Defect Prediction

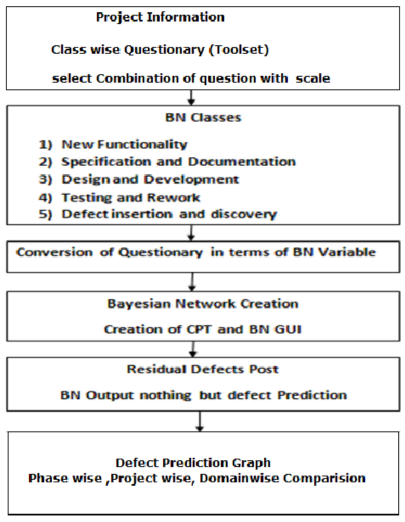

Project Information
Class wise Questionary (Toolset)
select Combination of question with scale
BN Classes
1) New Functionality
2) Specification and Documentation
3) Design and Development
4) Testing and Rework
5) Defect insertion and discovery
Conversion of Questionary in terms of BN Variable
Bayesian Network Creation
Creation of CPT and BN GUI
Residual Defects Post
BN Output nothing but defect Prediction
Defect Prediction Graph
Phase wise ,Project wise, Domainwise Comparision

Fig. 2. System Architecture BN Model

ample the Probability of Residual defects is simply the numerical Combination of Probability of all BN classes. In other cases, we can use standard statistical functions, too.
For BN Classes, the dynamic variable is shown with a bold boundary. The DBN is constructed with two nodes for each time indexed variable: the value in the previous time frame is the input node (i.e.pre Residual defect,prior probability) and it has no parents in the net [11]. The second node representing the value in this time frame is called the output node (here Post Residual defect ,posterior probability). The documentation is modeled, also, quality as a time varying quality attribute. It is usually being assumed that, documentation includes specification, which even in iterative developments is often prepared in one phase and implemented in a later phase. In this model, specification errors or bugs are considered as defects so a phase in which documentation is the main activity which lead to an important incremental change in documentation quality that is passed on to the next phase. Project manager and company has to make some important decisions in between the software development products course. One of the most important decision is the in time software product release. Very poor and biased decision may give problems to make compromise with quality which is in turn gives bad reputation to industry. Such decisions are often made on real ground and with basic information available rather than making on the plinth of more objective oriented and accountable in criteria. There are some uncertain parameters that make as stumbling block for development of software project. These are tools, personnel, development methods and testing strategies. These may give interference to do the delivery of a quality product in baseline budget and on time [10]. Each of these uncertain factors make always effects on SDLC right from requirement analysis to launching of product in the market [10]. In order to achieve qualitative software, some special attention and stress is need to apply for following three activities in particular:
• Defect prevention;
• Defect detection;
• Defect correction.
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 project in which project manager or Team members enter detailed information about every Project. Each project has five BN classes described below which has 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.

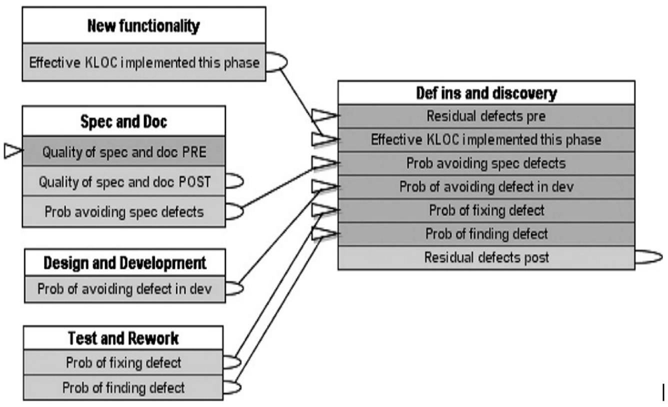

New functionality
Effective KLOC implemented this phase
Spec and Doc
Quality of spec and doc PRE
Quality of spec and doc POST
Prob avoiding spec defects
Design and Development
Prob of avoiding defect in dev
Test and Rework
Prob of fixing defect
Prob of finding defect
Def ins and discovery
Residual defects pre
Effective KLOC implemented this phase
Prob avoiding spec defects
Prob of avoiding defect in dev
Prob of fixing defect
Prob of finding defect
Residual defects post

Fig. 3. Objects in the phase BN

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 visible (**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

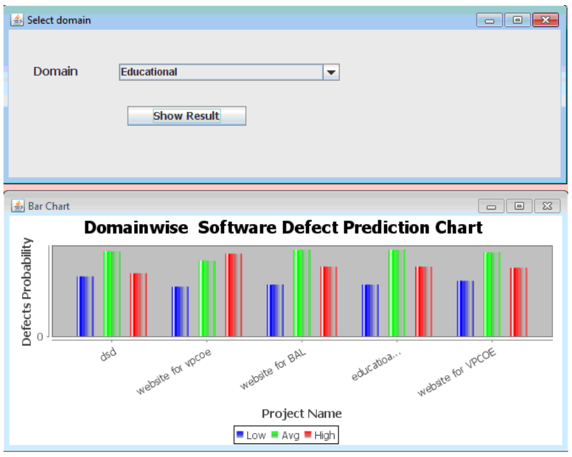


Fig. 9. Result: Domain based Comparison

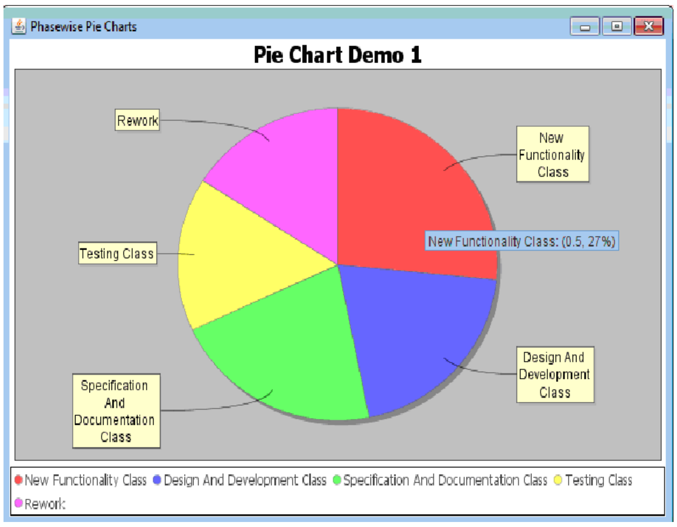
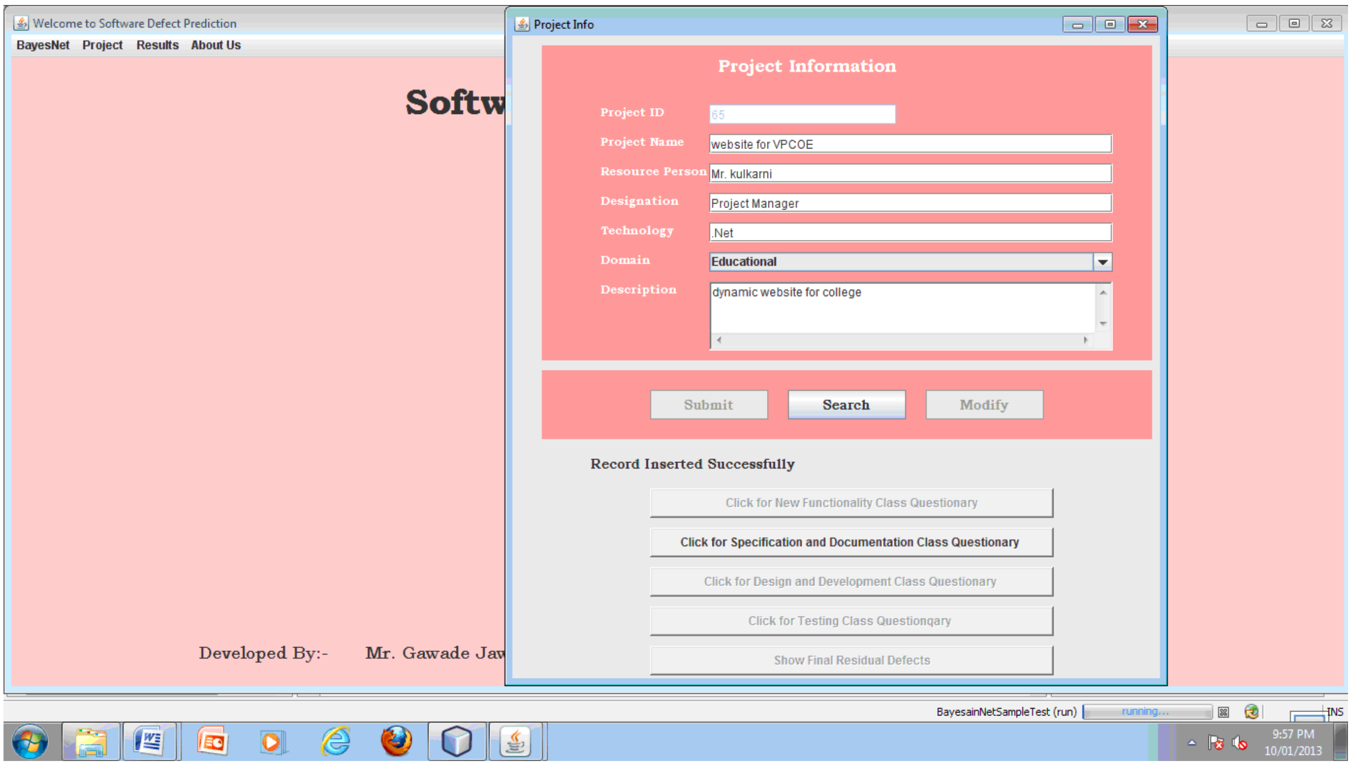

Phasewise Pie Charts
Pie Chart Demo 1
Rework
New Functionality Class
New Functionality Class: (0.5, 27%)
Testing Class
Design And Development Class
Specification And Documentation Class
New Functionality Class
Design And Development Class
Specification And Documentation Class
Testing Class
Rework

Fig. 10. Result: Phase based Comparison

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.



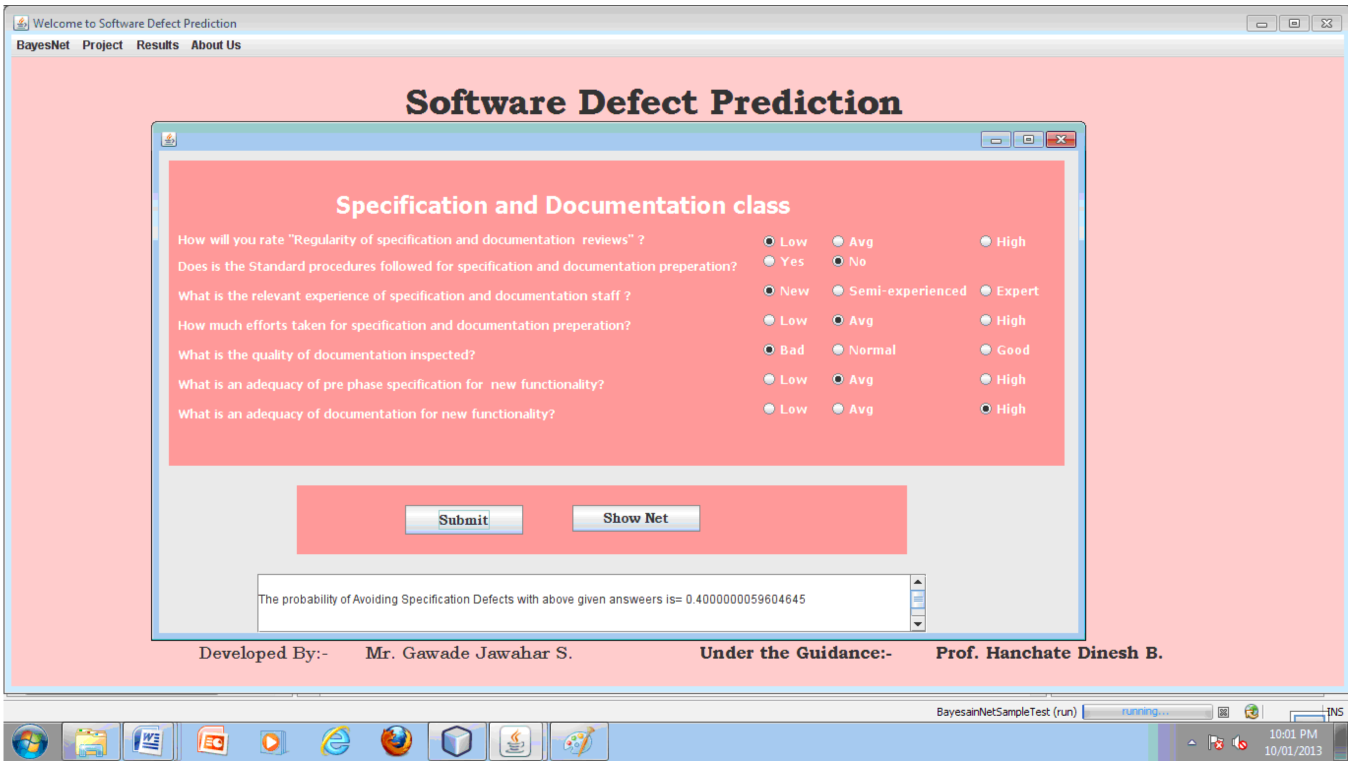
Project Information

Project ID	65
Project Name	website for VPCOE
Resource Person	Mr. kulkarni
Designation	Project Manager
Technology	.Net
Domain	Educational
Description	dynamic website for college

Record Inserted Successfully

Click for New Functionality Class Questionary
Click for Specification and Documentation Class Questionary
Click for Design and Development Class Questionary
Click for Testing Class Questionqary
Show Final Residual Defects

Fig. 4. Input: Project information

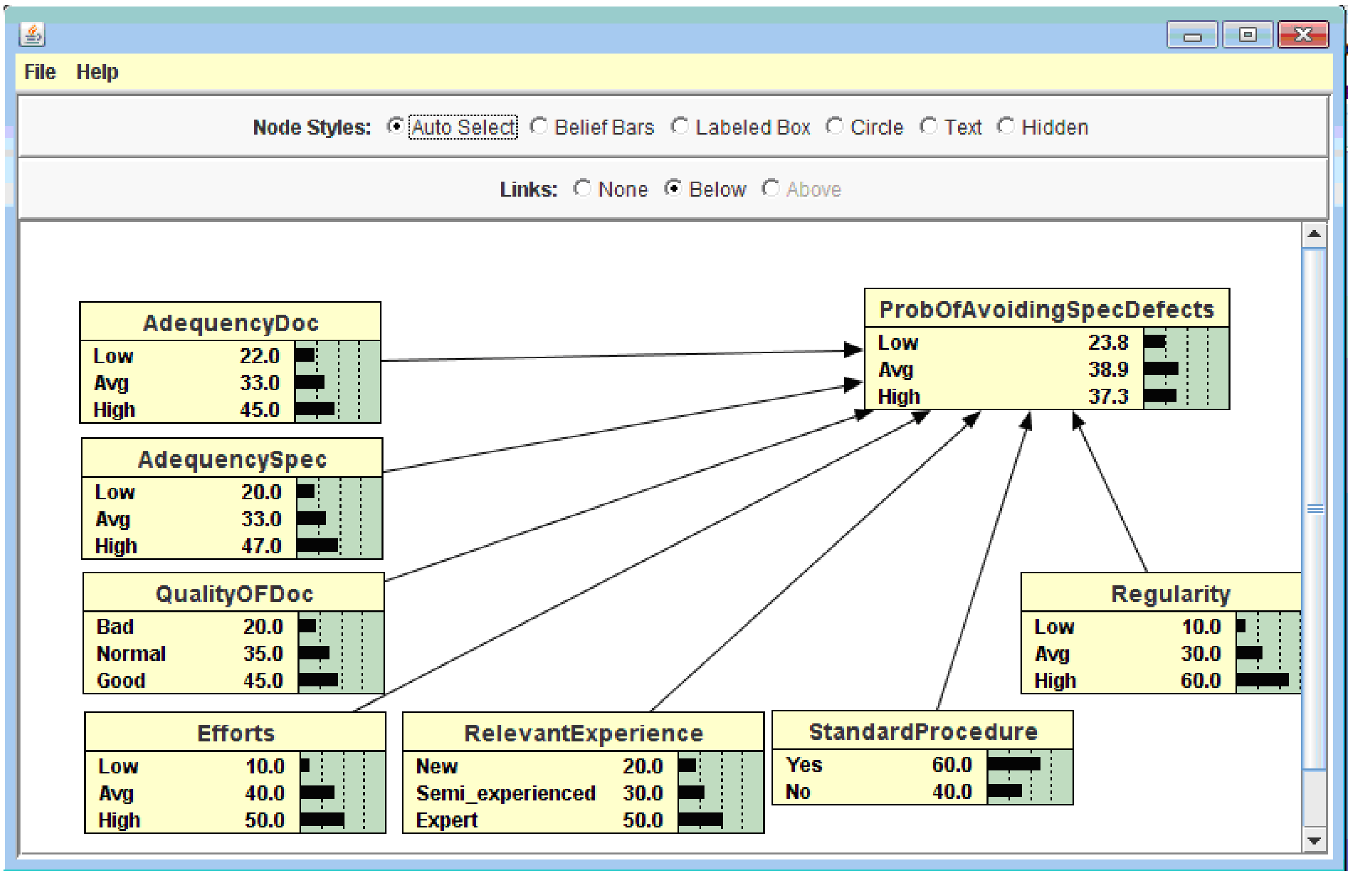


Specification and Documentation class

How will you rate "Regularity of specification and documentation reviews" ?	[✓ Low]	Avg	High
Does is the Standard procedures followed for specification and documentation preperation?	Yes	[✓ No]	
What is the relevant experience of specification and documentation staff ?	[✓ New]	Semi-experienced	Expert
How much efforts taken for specification and documentation preperation?	Low	[✓ Avg]	High
What is the quality of documentation inspected?	[✓ Bad]	Normal	Good
What is an adequacy of pre phase specification for new functionality?	Low	[✓ Avg]	High
What is an adequacy of documentation for new functionality?	Low	Avg	[✓ High]

The probability of Avoiding Specification Defects with above given answeers is= 0.4000000059604645

Fig. 5. Input: Specification Phase Questionnaires



Node Styles: [✓ Auto Select] Belief Bars Labeled Box Circle Text Hidden

Links: None [✓ Below] Above

AdequencyDoc	
Low	22.0
Avg	33.0
High	45.0

AdequencySpec	
Low	20.0
Avg	33.0
High	47.0

QualityOFDoc	
Bad	20.0
Normal	35.0
Good	45.0

Efforts	
Low	10.0
Avg	40.0
High	50.0

RelevantExperience	
New	20.0
Semi_experienced	30.0
Expert	50.0

StandardProcedure	
Yes	60.0
No	40.0

Regularity	
Low	10.0
Avg	30.0
High	60.0

ProbOfAvoidingSpecDefects	
Low	23.8
Avg	38.9
High	37.3

Fig. 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 [2]

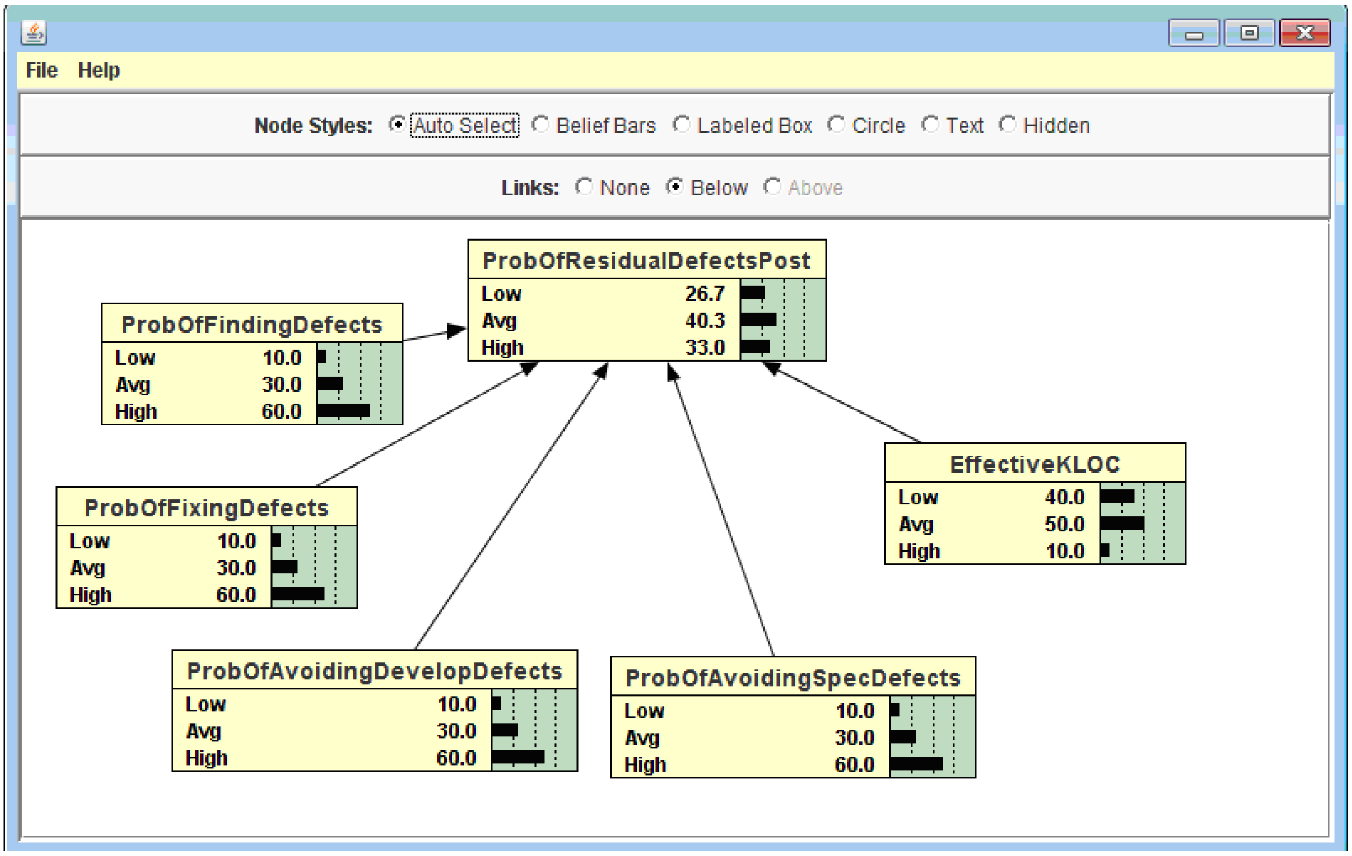
Figures 3 to 6 represents input to the model designed.

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



Node Styles: [✓ Auto Select] Belief Bars Labeled Box Circle Text Hidden

Links: None [✓ Below] Above

ProbOfResidualDefectsPost	
Low	26.7
Avg	40.3
High	33.0

ProbOfFindingDefects	
Low	10.0
Avg	30.0
High	60.0

ProbOfFixingDefects	
Low	10.0
Avg	30.0
High	60.0

EffectiveKLOC	
Low	40.0
Avg	50.0
High	10.0

ProbOfAvoidingDevelopDefects	
Low	10.0
Avg	30.0
High	60.0

ProbOfAvoidingSpecDefects	
Low	10.0
Avg	30.0
High	60.0

Fig. 7. Result: Bayesian Network for Residual Defects Post

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

[1] Agena. A critique of software defect prediction research. 2004.

[2] Norman Henderson Fenton and S.L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Company, 1997.

[3] Jensen F.V. *An Introduction to Bayesian Networks*. UCL Press, 1996.

[4] Dimitris Margaritis. Learning bayesian network model structure from data. May 2003.

[5] Richard Prewitt Michael Shannon, Geoffrey Miller. *Software Testing Techniques: Finding the Defects that Matter*. Charles River Media, 2005.

[6] Nielsen L. Neil M.Fenton, N. E. *Building large-scale Bayesian Networks: The Knowledge Engineering Review*. 2000.

[7] Netica. *Java Version of Netica API Norsys Software Corp*. Manual Version 4.18 and Higher.

[8] David Marquez Norman Fenton, Martin Neil. Using bayesian networks to predict software defects and reliability. "http://www.agenarisk.com/resources/white_papers",.

[9] Martin Neil Norman Fenton. A critique of software defect prediction research. *IEEE Trans. Software Eng.*

[10] Paul Krause Norman Fenton, Martin Neil. A probabilistic model for software defect prediction,. *For submission to IEEE Transactions in Software Engineering*.

[11] Rajat Mishra Norman HendersonFenton, Martin Neil. *Predicting software defects in varying development lifecycles using Bayesian nets*. Information and Software Technology, London, 2007.

[12] Roger Pressman. *Software Engineeing*. 2007.

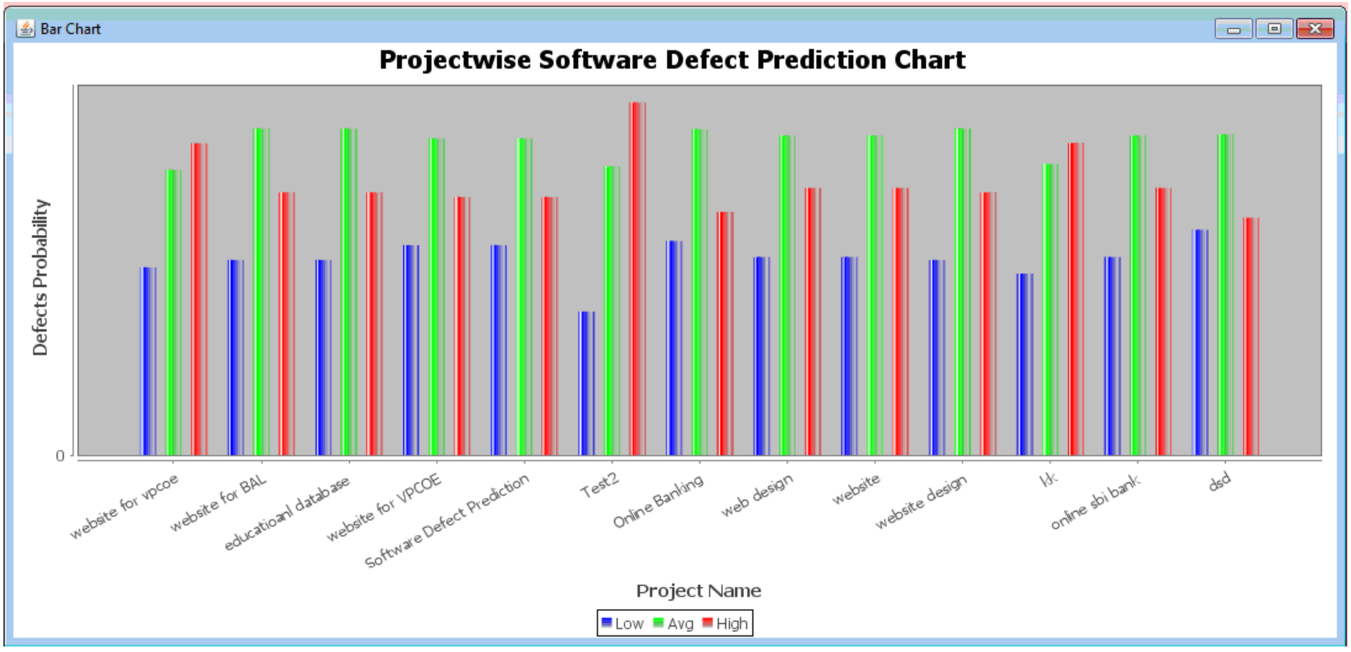


Fig. 8. Result: Project based Comparison

[13] N. Fenton R and M. Neil. A critique of software defect prediction research. *IEEE Transactions in Software Engineering*, 1999.

[14] Lukasz Radlinski. *Building Bayesian Nets for software defect prediction comarision of manual,semi- and fully-Atomated schemes*.

Jawahar Sambhaji Gawade B.E. Computer(2002-03); Network Admin (2004-06); Asst.Prof. since (2006- till date) SVPM's College of Engineering, Malegaon(Bk),Baramati,Pune., M.E. Computer (Sptr.2013) from VPCOE, Baramati, Pune. Confernce Convenor-2013, CCNA-CISCO certification-2003, Perceiving M.B.A. from SVPM's Institute of Management, Malegaon(Bk),Baramati,Pune, Network Engineer-2003-04 at ATSON Services, Rasta Peth, Pune.

Dinesh Bhagwan Hanchate Birth Place :- Solapur, B.E. Computer from Walchand College of Engineering, Sangli (1995), Lecturer in Gangamai College Of Engineering,Dhule (1995-96), Lecturer in S.S.V.P.S.s B.S.D. College Of Engineering,Dhule In Computer & IT deptt (1996-2005), M.Tech. Computer from Dr. Babasaheb Ambedkar Technological University, Lonere(2002-05), Currently Asst. Prof. Computer Engineering, former H.O.D. (Computer & IT) in Vidya pratishthans College Of Engineering, Baramati , currently doing research (SGGSs Institute of Technology and Engg, Nanded affiliated to SRTMU,Nanded) under the guidance of Dr. Bichkar R.S. ,G.H. Raisonis College Of Engineering and Management,Wagholi,,Pune.