DETECTION OF KERNEL-BASED MODULE AND DEFECT ANALYSIS USING MULTIVARIATE BAYESIAN METHOD

Dr.T.Rajagopalan[1],Bharath E[2]

[1]Dept. of Mathematics, University College of Engineering Ariyalur, Tamil Nadu, India, rajagopalant@rediffmail.com.

[2]Dept. of Computer Science and Engineering, University College of Engineering Villupuram, Tamil Nadu, India, bharath.elan@gmail.com.

ABSTRACT - Software project success is the key challenge. An automated approach to the incremental consistency checking of software design models. Analysis and Prediction of software module defects is the main focus for the engineering community. In this work, Multivariate Bayesian Prediction (MBP) Numerous product advancement organizations keep up their product software, and it is useful for the forecast of programming defects. In this work, Multivariate Bayesian Prediction (MBP) method has been utilized for finding the product faults before the testing procedure. This procedure causes us to decrease the cost of programming testing which diminishes the cost of the product venture. The machine learning-based prediction algorithms, manipulating the data, effort-aware prediction and empirical studies. The research community is still facing some challenges for building methods, and many research opportunities exist. The identified challenges can give some practical guidelines for both software engineering researchers in future software module and class defect prediction. A brief overview of some popular Bayesian reasoning methods is provided along with a justification apply to software module. In this approach to predict the class function and defects across projects even with heterogeneous metric sets. Our conclusion for this work is that, even when projects use different metric sets, it is possible to quickly transfer lessons learned about defect prediction.

INDEX TERMS- machine learning-based, MEBN, MBP, Bayesian Prediction Model

1. INTRODUCTION

Today, a huge corpus of software applications, which implement the entire spectrum of robot functionality, algorithms, and control paradigms, is available in robotic research laboratories and potentially could be reused in many different applications. The software has been proposed as a major discipline to manage the complexity of software systems from the high

abstraction levels and system wide perspectives. It is also well accepted that software manifest the earliest design decisions. On the one hand, the transition from waterfall development methods to agile methods has made software development more adaptable to suddenly changing conditions. On the other hand, sudden changes and the complexity of software have also caused software development to become a more difficult process to handle. Ways to see into a software process would be valuable.

The large codebase is a problem as the code will contain errors and require a lot of testing and code reviewing. On the one hand, code reviewing is an effective practice for finding errors in the early stages of development. On the other hand, even when using modern code review tools, it costs expert coders time that they could use to be effective in other ways. A proper amount of code reviewing balances the price of otherwise missed errors with the effort put into reviewing the code. Pinpointing the problem areas would be a way to make code reviews more effective.

Building high quality software with limited quality assurance budgets has become difficult. Various Software prediction models are used these days to learn fault predictors from software metrics. Software fault prediction, before the release of software helps in verification and validation activity and allocate the limited resources to modules which are predicted to be fault prone. Early and accurate fault prediction is a better approach for reducing testing efforts.

Reuse –oriented software base on reusable components and integrated framework for the composition of these components, that components may provide specific function such as word processing and spreadsheet. Type of software components that is used in reuse-oriented software process are Web services, Services standard are used for development these standard are available for remote. The routine use of existing solutions in the development of new systems is a key attribute of every mature engineering discipline. Software reuse is a state of the practice development approach in various application domains, such as telecommunications, factory automation, automotive, and avionics. Software Engineering has produced several techniques and approaches for promoting the reuse of software in the development of complex software systems.

2 RELATED WORKS

The method is proposed to extend the previously developed fuzzy rule-based Bayesian reasoning (FuRBaR) [1] approach to a wider context, where not only ambiguous estimates but also dependent relations associated with failures can be appropriately modelled. In this article

[2], to propose a novel approach to model the uncertain context by using ontology and context reasoning method based on Bayesian Network. Our context aware processing is divided into two parts: context modeling and context reasoning. The context modeling is based on ontology for facilitating knowledge reuse and sharing.

Augmenting existing [3], frameworks with a generalized hierarchical hybrid context reasoning engine (HyCoRE) could improve their reusability. As a step towards HyCoRE, we evaluate MEBN, a first order Bayesian network knowledge formalism, for suitability in data modeling and reasoning for pervasive computing contexts. In this paper [4] discussed Bayesian probabilistic reasoning can then be used to calculate the probability of particular choices leading to a successful design. In this way the most promising alternatives can be investigated first.

To provide theoretical and historical details on evidential reasoning using the chain rule [5]. Then we explore some questions about the relationship between Bayesian Networks and the functionality of a human brain as our last topic in Bayesian networks. The ways and processes [7] of human thinking developed by Psychologists and welcomed by computational experts produce the science of Artificial Intelligence. This process is called Knowledge Growing System which is Brain Inspired Cognitive Artificial Intelligence and can be used for information extraction.

The theoretical framework of cognitive informatics covers the Information-Matter-Energy (IME) model, the Layered Reference Model of the Brain (LRMB) [8], the Object-Attribute-Relation (OAR) model of information representation in the brain, the cognitive informatics model of the brain, Natural Intelligence (NI), and neuro informatics.

The control methods and strategy of [10] each control thinking process are discussed in this paper, the storage and mining of control experiences are analyzed. The test results are given in this paper, it is proved that the research content is right and could be used in the process control. It provides the basis for Human-like intelligent control to be used in complex system.

We provide a critical review of [13] this literature and the state-of-the-art. Most of the wide range of prediction models use size and complexity metrics to predict defects. Others are based on testing data, the “quality” of the development process, or take a multivariate approach.

The proposal [14] LLE-SVM model performs better than SVM model, and it is available to avoid the accuracy decrease caused by the data redundancy. Software defect prediction is a

kind of technology which can reveal the possibility whether a software system contains defects by analyzing the metric data of software.

The proposed methodology [15] helps in identifying modules that require immediate attention and hence the reliability of the software can be improved faster as higher priority defects can be handled first. Our goal in this research focuses to improve the classification accuracy of the Data mining algorithm.

One solution to this problem is developing [16] a systematic data collection procedure through standard guidelines that would be available to open community and thus enable reducing data collection bias. In this paper we present a tool demonstration that implements a systematic data collection procedure for software defect prediction datasets from the open source bug tracking and the source code management repositories.

This process provides [17] a rigorous method for establishing, maintaining, and reporting the correspondence between a system's requirement specification and the system's architecture, design components, modules, interfaces, test approaches, and test data throughout the software life cycle.

The VRI software maintains [18] a database of information from potential volunteers and potential opportunities, and provides mechanisms to help the involved parties, as well as GLVB personnel, identify and follow up on good matches.

The course is based on software [19] modeling using UML, deployment of design patterns, code generation especially for embedded systems and using model driven architecture principles. This basic knowledge is applied to a programming project in robotics. Starting with a Lego NXT robot without Lego software the ARM 7 processor is addressed directly by the developed software.

This paper proposes a set of [20] measures for capturing a number of internal attributes of software specifications written with Petri nets, a well-known formal technique for modeling concurrent systems.

3. IMPLEMENTATION OF PROPOSED WORK

Multivariate Bayesian Prediction (MBP) technique is a machine learning technique. The MBP technique has already been used in cost estimation in the field of software Engineering but has never been explored in software fault prediction. The main advantage of the BR technique is

that it consumes less memory space and provides better accuracy than all the previous techniques used in the software fault prediction model.

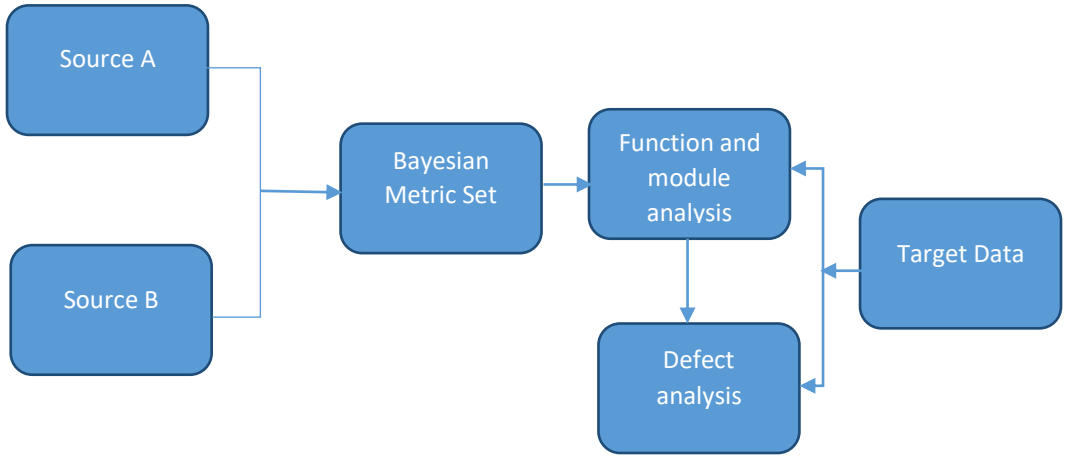

Fig 1: Proposed Method Block Diagram

Defects in system software lead to foremost difficulty in the software. A lot of software systems are sent to the clients with unnecessary defects. Testing is one of the most important approaches for finding the defect prone parts of the system. Software quality can be measured with various attributes like fault thickness, normalized rework, reusability, portability and maintainability etc.

Bayesian Prediction Model

Risk-based testing, which uses predicted risks to guide the test process, is employed to select test cases. To this end, risks have so far mainly been estimated ad hoc, but not systematically predicted on the basis of the defect history and defect costs. To present a novel approach to risk-based test selection, which employs a comprehensive and versatile Bayes risk model taking defect probabilities and costs into account. It enables the prediction of a risk decrement that could potentially be used for test selection.

Procedure Model Construction for Defect Prediction

Input: Predictor variables, namely, defect density *g*, defect velocity *v*, and defect introduction time *t*

Output: Predicted number of defects

pre-process datasets ($M_{1\text{-}n}$)

determine (*g*, *v*, *t*);

for*i* D 1 to *n***do**

for*j* D 1 to *n* - 1 **do**

if*M*(*g*, *v*, *t*) could not be determined **then**

forevery (M_{1-n}), recheck*M*(*i*, *j*) to determine

(*g*, *v*, *t*) **do**

if*M*(*g*, *v*, *t*) have been determined **then** build a model *f* (*g*, *v*, *t*)

end if

end for

end if

Divide *M*(*i*; *j*) into two parts, 80% and 20%, and store in variables **Tdata**and **Vdata**, respectively

Tdata training (80%)

Vdata validation (20%)

train model with *M*(*i*; *j*) (80%)

validate model with *M*(*i*; *j*) (20%)

if*M*(*i*; *j*) validation D successful **then**

determine prediction outcome

else

Clean and pre-process dataset (M_{1-n})

Again

End if

End for

End for

End procedure

Software engineering researchers analyze programs by applying a range of test cases, measuring relevant statistics and reasoning about the observed phenomena. Though the traditional statistical methods provide a rigorous analysis of the data obtained during a program analysis, they lack the flexibility to build a unique representation for each program.

Result and discussion

The implementation was carried out through visual studio framework 4.0 with SQL server authentication.

Analysis of time complexity

The time complexity parameter has been calculated by using a database and the request queries are the input of the listed algorithm based on the number of users and the execution time of the individual algorithms are represented below:

QA= query analytics

T= trust

A= accuracy

DA= No of database access

N=no of user

Time = (QA+T+A)/(DA +N)

The figure given below shows the Time complexity by different comparisons as follows.

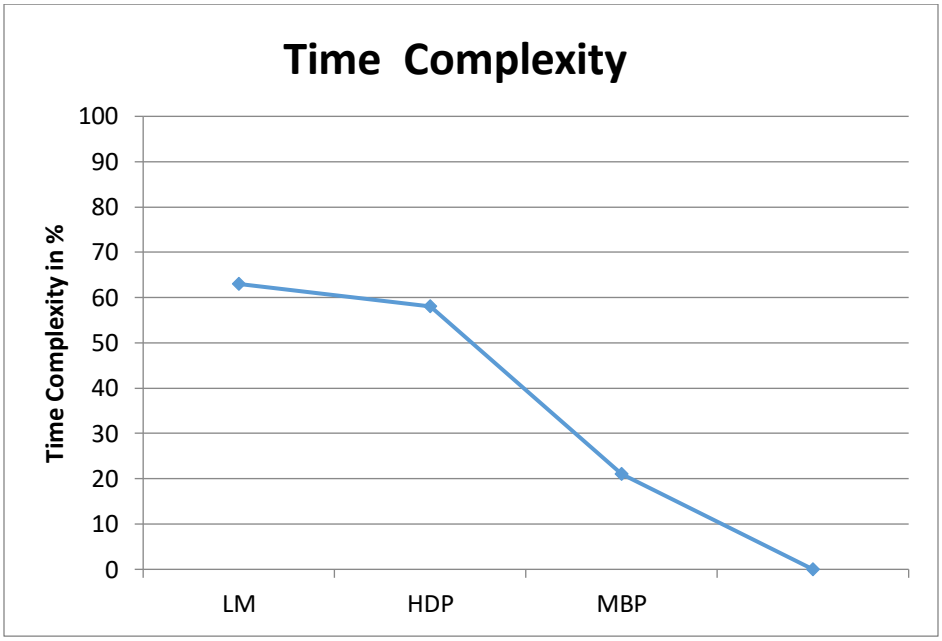

Fig 2: Time Complexity

Figure 2 Shows the comparative result on time complexity produced by various methods and shows clearly that the proposed method has produced less time complexity.

Prediction accuracy

Prediction accuracy is a measure to predict the outcome values for previously unseen data it is also called as out of sample accuracy. For measurable data type, we noted that if all the data are self-governing to each other, it is impossible to compute the true values and get the accuracy of the data without enough prior information. Since we could not often obtain sufficient prior

knowledge and many tuples may describe the same entity, we could use the entity resolve technology.

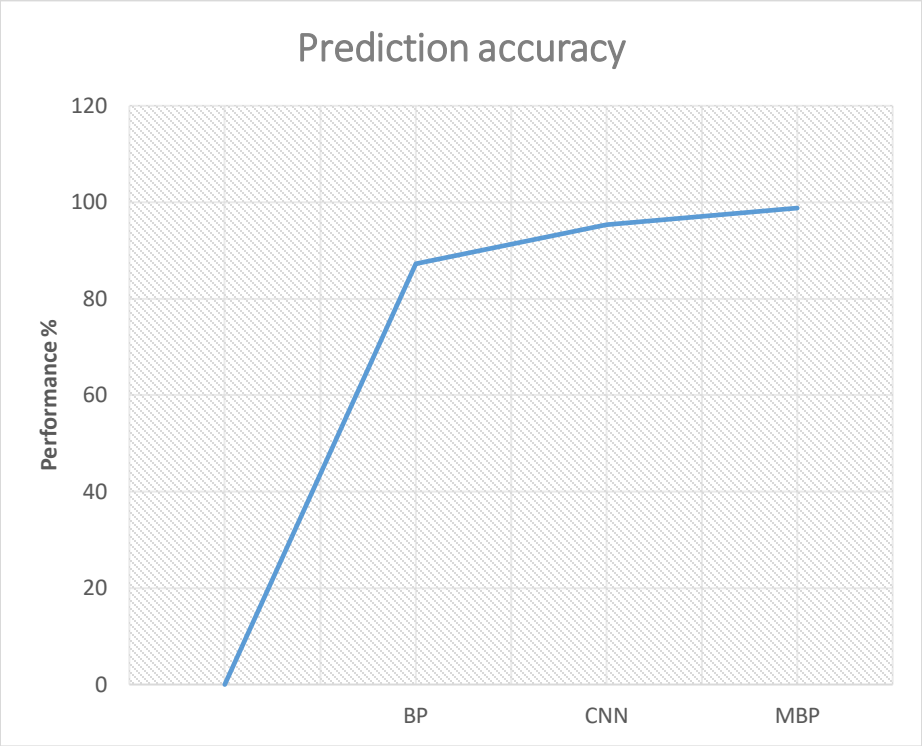

Fig 3: Prediction accuracy

.

4. CONCLUSION

Multivariate Bayesian Prediction (MBP) algorithm is proved to be the best algorithm. Neural Network is based upon machine learning approach. As a result, it is found that machine learning models are importantly used and provide superior results. The time complexity parameter has been calculated by using a database and the request queries are the input of the listed algorithm based on the number of users and the execution time of the individual algorithms. By this technique we will reduce the time complexity and increase the prediction accuracy.In future, we will explore the feasibility of building various prediction and recommendation models using heterogeneous datasets. Some other training algorithms may be tried to raise the accuracy level for finding the software faults at an early stage of software development life cycle. By using class level metrics, more studies can be conducted on fault prediction models.

REFERENCE

[1]. Zaili YANG, Stephen BONSALL, Jin WANG, A Fuzzy Bayesian Reasoning Method to Realise Interactive Failure Analysis, IEEE 2009, pg.no: 403 – 406.

[2]. Kwang-EunKo and Kwee-Bo Sim, Development of Context Aware System based on Bayesian Network driven Context Reasoning Method and Ontology Context Modeling, IEEE 2008, pg.no: 2309 – 2313.

[3]. Bridget Beamon, Evaluation of First Order Bayesian Networks for Context Modeling and Reasoning, IEEE 2010, pg.no: 867 – 868.

[4]. John Jones, Yang Xiang and Stefan Joseph, Bayesian Probabilistic Reasoning in Design, IEEE 1993, pg.no: 501 – 504.

[5]. CAO Yonghui, Study of the Bayesian Networks, IEEE 2010, pg.no: 172 – 174.

[6]. David P. Menzer, An Application of Bayesian Reasoning to Improve Functional Test Diagnostic Effectiveness, IEEE 2002, pg.no: 711 – 719.

[7]. AdangSuwandi Ahmad, Brain Inspired Cognitive Artificial Intelligence for Knowledge Extraction and Intelligent Instrumentation System, IEEE 2017, pg.no: 352 – 356.

[8]. Yingxu Wang, Cognitive Computing and Machinable Thought, IEEE 2009, pg.no: 6 – 8.

[9]. Jong-Hwan Kim, Seung-Hwan Choi, In-Won Park, and SheirAfgenZaheer, Intelligence Technology for Robots That Think, IEEE 2013, pg.no: 1 – 1.

[10]. Peijin Wang, Analysis and Implementation of Human Control Thinking Process, IEEE 2006, pg.no: 2146 – 2150.

[11]. Cindy Mason, The Logical Road to Human Level AI Leads to a Dead End, IEEE 2010, pg.no: 312 – 316.

[12]. Sara Ayoubi, NouraLimam, Mohammad A. Salahuddin, NashidShahriar, RaoufBoutaba, Felipe Estrada-Solano, and Oscar M. Caicedo, Machine Learning for Cognitive Network Management, IEEE 2018, pg.no: 158 – 165.

[13]. Norman E. Fenton, Martin Neil, A Critique of Software Defect Prediction Models, IEEE 1999, pg.no: 675 – 689.

[14]. Chun Shan, Boyang Chen, Changzhen Hu, JingfengXue, Ning Li, Software Defect Prediction Model Based On Lle And Svm, IEEE 2014. Pg.no: 1 – 5.

[15]. K.PUNITHA, Dr. S. CHITRA, Software Defect Prediction Using Software Metrics - A survey, IEEE 2013, pg.no: 555 – 558.

[16]. GoranMauˇsa, TihanaGalinacGrbac, BojanaDalbelo, Software Defect Prediction with Bug-Code Analyzer - a Data Collection Tool Demo, IEEE 2015, pg.no: 1-2.

[17]. David Sharon, A Complete Software Engineering Environment, IEEE 1997, pg.no: 123 - 125.

[18]. Panagiotis K. Linos, Chris Bailey-Kellogg, Service Learning in Software Engineering and Maintenance, IEEE 2003, pg.no: 336.

[19]. Marianne von Schwerin, Software Engineering in a Nutshell for Electrical Engineering Students, IEEE 2014, pg.no: 788 – 793.

[20]. SandroMorasca, Measuring Attributes of Concurrent Software Specifications in Petri Nets, IEEE 06 August 2002, pg.no: 100 – 110.

[21]. Richard B. Kieburtz, Laura McKinney, Jeffrey M. Bell, A Software Engineering Experiment in Software Component Generation, IEEE 1996, pg.no: 542 – 552.

[22]. Nee1 Madhav, SriramSankar, Application of Formal Specification to Software Maintenance, IEEE 1990, pg.no: 230 – 241.

[23]. Professor Paul Layzell, Addressing the Software Evolution Crisis Through A Service-Oriented View of Software: A Roadmap for Software Engineering and Maintenance Research, IEEE 2001, pg.no:5.

[24]. Ke Liang, Juan Guo and Jinghua Wang, On-Board Software Maintenance for Manned Spacecraft, IEEE 2014, pg.no: 235 – 239.

[25]. Denis Kozlov, JussiKoskinen, JouniMarkkula, MarkkuSakkinen, Evaluating the Impact of Adaptive Maintenance Process on Open Source Software Quality, IEEE 2007, pg.no: 186 – 195.