# Software Quality Assurance Development Using Bayesian Graphical Model and Safe Growth Model

**Dejan Chandra Gope** dejan.gope23@gmail.com

**Dr. Mohammod Abul Kashem** drkashemll@duet.ac.bd

Department of Computer Science and Engineering
Dhaka University of Engineering and Technology (DUET)
Gazipur-1700, Dhaka, Bangladesh.

**ABSTRACT**: Software quality assurance is a planned and systematic approach to ensure that software processes and products confirms to the established standards, processes, and procedures. The goals of software quality assurance are to improve software quality by appropriately monitoring both software and the development process to ensure full compliance with the established standards and procedures. There are several models for software quality assurance, such as the ISO/IEC 90003, and the capability maturity model integration. As the software in today's systems grows larger, it has more defects, and these defects adversely affect the safety, security, and reliability of the systems. Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software. Quality is conformance to product requirements and should be free. This research concerns the role of software Quality. Software reliability is an important fact of software quality. It is the probability of failure-free operation of a computer program in a specified environment for a specified time. In software reliability modeling, the parameters of the model are typically estimated from the test data of the corresponding component. This research describes a new approach to the problem of software testing. The approach is based on Bayesian graphical models and presents formal mechanisms for the logical structuring of the software testing problem, the probabilistic and statistical treatment of the uncertainties to be addressed, the test design and analysis process, and the incorporation and implication of test results.

**Keywords-**: Bayesian Data Analysis, Probabilistic Reasoning, Software Testing, Program Analysis, Software Reliability, Uncertainty Analysis, Bayesian Graphical Model (BGM), Test Design, MEP, MC, TTF.

## 1 INTRODUCTION

The theory of BGMs has led to many new applications of uncertainty modeling, in particular, to complex problems where a large number of factors contribute to overall uncertainty. BGMs derive from Bayesian statistical methodology, which is characterized by providing a formal framework for the combination of data with the judgments of experts such as software testers. For the application area of software testing, we demonstrate how the problem should be structured and how the resulting models may be used. We illustrate the methodology with case studies arising from applying the approach to large scale software testing problems for a major UK company.

In software reliability modeling, the parameter of the model is typically estimated from the test data of the corresponding component. However, the widely used point estimators are subject to random variations in the data, resulting in uncertainties in these estimated parameters. Ignoring the parameter uncertainty can result in grossly underestimating the uncertainty in the total system reliability to apply the models for predicting the reliability of the component; the parameters of the models need to be known or estimated. Field data or data from components with similar functionalities are usually available to help estimate these parameters, but the estimators are subject to random variation because they are functions of random phenomena. Parameter uncertainty arises when the input parameters are unknown. Moreover, the reliability computed from the models, which are functions of these parameters, is not sufficiently precise when the parameters are uncertain.

## 2 RELATED WORK

Sofware reliability have attracted attention from statistical researchers. However, much of this work attempts to fit problems related to software reliability within existing mathematical frameworks rather than attempting carefully to model the actual uncertainties occuring in software testing and the process of learning from tests. We make case studies central to the development of the methods described in this paper as the emphasis of our approach is in modeling the actual testing process and thus contributing to better testing.

Smidts and Sova present an approach for software reliability quantification, placing the functional architecture of the software centrally in their model. They suggest that their model "encourages a testing philosophy directed toward the triggering of failure modes and removal of related faults," but they do not provide further guidelines for testing at the input partition level. It would be interesting to consider if such, or related, models could be used in the process of creating BGMs for software testing.

Frankl et al. identify two main goals in testing software: to achieve adequate quality (debug testing) and to assess existing quality (operational testing). The objective for debug testing is to probe software for defects so that these can be removed. The objective for operational testing is to gain confidence that the software is reliable. They examine the relationship between these testing goals via a probabilistic analysis in which the effectiveness of testing is based on the reliability of a program after testing. Both approaches are based on subjective arguments: Debug testing relies on insights on where faults are likely to be; operational testing depends on knowledge and assumptions on operational profiles. Both approaches have advantages, depending on the practical situation, and both depend on partitioning of the input space. The BGM approach also requires partitioning the input space. Our partitioning is driven by focusing on differing software actions (SAs), which we regard as essential to test complex software. Frankl et al. carefully discuss the difficult problem of defining "faults" that are responsible for failures and suggest avoiding the term "bug" because of its often vague definition. They conclude that a formal treatment of "faults" is not available and suggest using "failure regions" of the input space, where one such region is a set of failure points that is eliminated by a program change. We stay close to this in our analysis by probabilistically tracking back an observed failure in the graphical model to see to which specific SA the cause of failure is likely related, to guide attempts to fix the fault. Detailed operational testing enables a statistical analysis, when based on a large number of tests, of the reliability of the software in operation. While we support such testing whenever possible, resource constraints for our applications, together with vague knowledge about operational profiles, often prevent us from applying such testing. However, some aspects of operational profiles are reflected in utilities which influence the design of test suites.

Regression testing is the process of testing a program after changes have been made to it to ensure that the changes have been effective and have not introduced further faults. For such testing, there already exists a set of prior test cases. It is not usually an option to rerun all of

these tests, so some method for regression test selection needs to be devised. Rothermel and Harrold evaluate current regression test methods. All of the methods analyzed are based on information about the source code before and after modification and yet none of the methods attempt to capture expert knowledge about the software and the tests. Regression testing is not explicitly addressed in this paper, but is deferred to a subsequent report in which we will demonstrate formal logical and probabilistic mechanisms for regression testing within the BGM approach and which enables a routine and automatic treatment of regression testing.

Many statistical methods have been suggested or used in attempts to create better software. Burr and Owen describe the possible use of methods from classical statistical quality control, although the adaptation to software problems is rather vague. Our first case study was far too complex to allow any standard classical statistical methods to be used in a straightforward fashion and this appears to be typical for testing problems. Singpurwalla and Wilson give overviews of an active area of statistical research in software reliability. Mostly, these are contributions to the theory of stochastic processes, linking reliability metrics to assumed behavior of processes with which failures occur, both while testing and in operation. They also discuss testing aspects related to such assumed models. While such approaches are potentially interesting, it seems that direct application is currently only possible to software of rather restricted complexity.

BGMs, also called Bayesian belief networks (BBN), have been applied to different problems on software quality. Neil and Fenton use them to predict software quality, taking into account a diversity of factors such as effort and complexity of design, skills of people involved in the process of development and testing, and costs. While this is an interesting approach to give an overall idea of the density of defects in a piece of software, these BBNs are not aimed directly at assisting testers. The most typical use of BBNs in this application area is in inferring models for reliability from large databases. The SERENE project (http://www.hugin.dk/serene/) presents interesting applications of BBNs in the area of safety and risk evaluation and some of the work within this project also takes software into account albeit without actual support at the test design level. Our usage of BGM rather than BBN reflects the terminology used in the wider statistics literature and helps to emphasize that our BGMs model testers' judgments and are not the results of inferring models from large databases.

## 3 SOFTWARE RELIABILITY AND ITS PROBLEM

Reliability is usually measured based on probability theory and, in general, mathematical statistics are used to estimate these probabilities. Reliability engineering is an important aspect of many system development efforts and consequently there has been a great deal of research in the software based systems. One important activity included in reliability engineering is reliability prediction. To control a complex industrial process, many heterogeneous components are needed to work together with highest reliability. The integration of heterogeneous components into mechatronics systems requires broadening the concepts and cooperation between different technical disciplines involved to develop a common conception of the future product and come up with an optimized solution. So the complexity of mechatronics system is to be broken by extracting the basic fundamental function of the different units. So, it is needed to abstract the reliability related peculiarities from each discipline and unify them in a way suitable for mechatronics as whole. The reliability definition stresses four elements namely i. probability ii. Intended functions iii. Time iv. Operating conditions.

The numerical evaluation of reliability is based on the total duration of failures or the frequency of failures. So, most benefit from the use of reliability can be achieved at the early phases of the design of programmable mechatronics system to identify the adequate

confidence about the systems and its components. The reliability analyses of various components involved in software based systems can be broadly classified in to
a. Software component reliability.
b. Interface and networking reliability.
c. Software controlled hardware reliability.

It has been noted that reliability of the large scaled electro-mechanical systems should be properly evaluated based on the subsystems reliability.
Definition for software "Set of programs, procedures and its related documentation for delivery to a customer. Software Quality deals with "fitness of Use" and "Satisfies user requirement". IEEE defines software quality as a software feature or Characteristic used to assess the quality of a system. Software Quality is measured in different ways, using internal parameters, and external attributes. Software reliability is Quality factor. Parameter definition is very important in Reliability. System level improved reliability factor is hard to find.

## 4 LITERATURE SURVEY

Software reliability models consider different elements of the software project, such as the specification & codification of the programs, or information about the fault detection & correction process. They are usually based on characteristics of the testing activity, and are classified according to the assumptions that each model does in its formulation. The parametric and traditional models assume in their analytic formulation a predetermined behavior where some parameters have to be adjusted, fitting the curve to the failure data. Their parameters are explicitly defined in the model, and have a physical interpretation. To adjust the parameters of the parametric models, there is always a set of assumptions that may not be suitable for most cases. The influence of external parameters and other peculiarities of a model can be eliminated if we have a model that is able to evolve itself based on the failure data collected during the initial test phase. Although they include parameters in their analytical formulation, they are also known as non-parametric reliability models because these parameters do not have a physical interpretation.
Reliability is frequently related to the probability of a failure occurring in the operational use of the system. Software systems are usually not considered to wear out with time. The only external factors that determine the output of a program are the input data. Faults in the software are essentially originated by human mistakes, such as a wrong specification, or instruction in the code. When a failure occurs, faults should be located & removed, so the system reliability tends to increase. Software reliability models are used to describe this process. They allow the estimation or prediction of the present or future reliability of a system.
Software qualities well known models are McCall, Boehm, FURPS .The following sections will discuss briefly on models:

### 4.1 The McCall model (1977)

The McCall model is constructed using tree-like fashion. This model holds quality factors usability which will be quantified. In this model Quality factors are not directly measured and set of metrics is needed to develop relationship.

### 4.2 The Boehm model (1978)

The Boehm model represents a hierarchical structure of characteristics, each of which contributes to total quality. Utility is broken in to different levels. This model is not considered for higher level.

### 4.3 The FURPS model (1987)

Hewlett-Packard developed a set of software quality factors that make up its name FURPS.It takes Functionality, Usability, Performance and supportability. Disadvantage of this model is that it does not take into account the software product's portability.

**4.4 The Systemic Quality Model (2003)** The systemic model is differed from the previous model mentioned above. This model is developed by identifying the relationship between product-process, efficiency-effectiveness and user-customer to obtain global systemic quality. The model proposed focuses on product quality, which includes efficiency and effectiveness and the concept of systemic global quality. The disadvantages of this model are that it does not cover the user requirements and conformant aspects. Analysis done on the different models demonstrates that different quality characteristics associated with these different models.

## 5 QUALITY SAFE GROWTH MODEL

In semi-supervised learning, decision rule is to be learned from labeled and unlabeled data. In this framework, we motivate minimum entropy regularization, which enables to incorporate unlabeled data in the standard supervised learning. Our approach includes other approaches to the semi-supervised problem as particular or limiting cases. A series of experiments illustrates that the proposed solutions benefit from unlabeled data. Another characteristic challenge in software testing and reliability is the lack of available failure data from a single test, which often makes modeling difficult. This lack of data poses a bigger challenge in the uncertainty analysis of the software reliability modeling. The existing semi-supervised learning techniques are all not very effective for our case of lack of failure date.

### 5.1 Objective

The objective is to quantify the uncertainties in the software reliability model of system with correlated parameters. Challenges in software reliability is the lack of available failure data from a single test, which often makes modelling difficult. Using that data poses a bigger challenge in the uncertainty analysis of the software reliability modelling. For achieving good quality, we use reliability model using Bayes approach with TQM.

### 5.2 RF Approach

RF (Reliability Factor), Reliability is the probability of a device performing its purpose adequately for the period intended under the given operating conditions. The definition brings into focus four important factors namely,

i. The reliability of a system is expressed as a probability.
ii. The system is required to give adequate performance.
iii. The duration of adequate performance is specified.
iv. The environmental or operating conditions are prescribed.

Reliability is usually measured in terms of probabilities. The theory of Bayesian statistics is a well established and the method has been applied in various areas including software, automation systems, medical diagnosis, geological explorations etc.
In the software based system, the uncertain variables are associated to each component where the uncertainty is expressed by probability density. The probability density expresses our belief or confidence in the various possible outcomes of variable. This probability depends conditionally on the status of other component based on different input.
The probabilities are estimated by means of statistical methods. Various statistical methods are available for estimating the reliability of the system, but these methods are not suitable for estimating the reliability of software based system, because, these methods have not

considered the uncertainties of unknown parameters. The Bayesian statistical method considers the uncertainties of unknown parameters. So, Bayesian model is mainly used for estimating the reliability of software-based systems. This method is also used to predict the future of the system by the observed information.
Reliability is the probability that a system will operate without failure for a given time in a given environment. Note that reliability is defined for a given environment.
Since the test and operations environments are generally different, model results from test data may not apply to an operations environment. The "given time" in this definition may represent any number of actual data items, such as number of executions; number of lines of code traversed, or wall clock time. The use of a model also requires careful definition of what a failure is. Reliability models can be run separately on each failure type and severity level. Models have been developed to measure, estimate and predict the reliability of computer software. Software reliability has received much attention because reliability has always had obvious effects on highly visible aspects of software development: testing prior to delivery, and maintenance. Early efforts focused on testing primarily because that is when the problems appeared. As technology has matured, root causes of incorrect and unreliable software have been identified earlier in the life cycle. This has been due in part to the availability of results from measurement research and/or application of reliability models.

Step 1. Take a prior factor p(a) with respect to a certain parameter, given a set of newly observed data.
Step 2. Using the newly observed data, estimate the parameter ‘f’.
Step 3. Derive g=∫ f’/4 by using that formulae.
Step 4. Compare the values ‘f’ and ‘g’.
Step 4. If (f < g) or (f > g), the adjustment should be triggered. The estimated parameter ‘E’ is located at the extreme tails of the prior distribution.
Step 5. Adjust the degree factor d* for filtering and then recalculating the prior distribution and then repeat Step 1.

### 5.3 Measure Information

It can be used to approach physical systems from the point of view of information theory, because the probability distributions can be derived by avoiding the assumption that the observer has more information than is actually available. Information theory, particularly the definition of information in terms of probability distributions, provides a quantitative measure of ignorance (or uncertainty, or entropy) that can be maximized mathematically to find the probability distribution that is maximally unbiased Entropy.
If any of the probabilities is equal to 1 then all the other probabilities are 0 and we then know exactly which state the system is in. Since probabilities are used to cope with our lack of knowledge, and since one person may have more knowledge than another, it follows that two observers may, because of their different knowledge, use different probability distributions. In this sense probability, and all quantities that are based on probabilities, are subjective.
`Our uncertainty is expressed quantitatively by the information which we do not have about the state occupied. This information is

$$K= \sum_{i,j} p\,(Ti+Tj)\,\log 2\,(1/p\,(Ti) + j\,(Ti)) \qquad (1)$$

$$L=\sum_{i,j} (p\,(Ti+Tj)*g\,(Ti+Tj) \qquad (2)$$

Information is measured in bits because we are using logarithms to base 2. One person may have different knowledge of the system from another, and therefore would calculate a different numerical value for entropy. The Principle of Maximum Entropy is used to discover

the probability distribution which leads to the highest value for this uncertainty, thereby assuring that no information is inadvertently assumed.

### 5.4 Reliability Model for All Modules of a System

The system reliability can be evaluated using the architecture and relationship of the components.

The Markov model, SPN, fault tree analysis, and reliability block diagram are some popular tools for evaluating the system reliability, given the parameters of its contained components.

For example, a Markov chain is characterized by its state space, together with the transition Probabilities over time between these states. Usually, there are four steps to construct and solve the Markov chain models:

1. Set up the Markov chain model.
2. List the Chapman-Kolmogorov equations.
3. Solve those equations to obtain state probabilities.
4. Obtain the reliability by summing up the probabilities of those reliable states.

Choose prior distributions based on previous knowledge either the results of earlier studies or non-scientific opinion.

Modern Parametric Bayesians and the normal model with unknown mean and variance

• Consider r and t2 are unknown random variables.

Bi ~ n(r , t2) (3)

• we used the definition of conditional probability, so

p (r ,(IL+t2) = p(r|(IL+t2)p(IL+t2)? (4)

• This is a conjugate distribution for the normal distribution with unknown mean and variance; the posterior distribution will also be normal-Inv-X2.

In the previous sections, we have analyzed the parameter uncertainty of reliability model for one component based on Entropy and Bayes. Complicated software contains multiple modules. Many tools can be implemented to evaluate the system reliability, given the parameters of its contained components, such as the Markov models, Bayesian Network, Graph Theory, and Fault-Tree Analysis. Regardless of the tools used, the system reliability is a function combining the parameters of its components. As a result, the uncertainties in the parameters affect the whole system reliability. The purpose of this section is to study and quantify the uncertainty in the reliability of the complex system due to the uncertainty of the parameters in the numerous components of the system. Some general assumptions of the system-level analysis are listed as follows:

1. Consider system contains different modules.
2. Each module has its own reliability model and let j denote the set of parameters for the ith module's model, where j = 1; 2. . . K.
3. For each module, the probability distribution of its model's parameters is known, which can be derived using formula2 and 4.
4. The failures of different components are'd' independent, the system reliability can be Calculated by the function of modules Parameters as d = f (v1, v2,……, vj).

Based on the above assumptions, an MC simulation is presented for generally analyzing the uncertain system reliability. It is difficult to use analytic methods for combining the distributions of numerous parameters to derive the probability density function of the system reliability, especially for complicated systems with complex architecture and many components. Hence, the MC simulation becomes a practical way to make the uncertainty analysis of the complicated system tractable. Algorithm 1 provides a general MC approach for the uncertainty analysis in a complicated system.

### 5.5 Infromation Validation

The posterior distributions validate the posterior of modules by using the following method:

- Calculate the mean values of the posterior distributions and obtain parameters.
- Using, these parameters can be applied to the test.
- Compare the predicted TimeToFailures with the real observed TimeToFailures, by calculating the mean square error. M (t) =b(1-exp(-¢(t)), a>0 (5)
- If the mean square error m(t)>d(Threshold value)is then the model does not fit the observed data.
- The adjustment (such as trying another model or further filtering the subjective information) should be applied.

| Model | | |
|---|---|---|
| **Failure Number** | **Failure Interval Length** | **Day of Failure** |
| **1** | **5** | **1** |
| **2** | **73** | **1** |
| **3** | **141** | **1** |
| **4** | **491** | **5** |
| **5** | **5** | **5** |
| **6** | **5** | **5** |
| **7** | **28** | **5** |
| **8** | **138** | **5** |
| **9** | **478** | **9** |
| **10** | **325** | **9** |

Table 1: Frequency Table

### 5.6 Safe Growth Model Quality Values

| k | L | p(r ,(IL+t2) | m(t) | Rank | RealibiltyValue |
|---|---|---|---|---|---|
| **0.981309** | **0** | **0** | **662.0769** | **242.4507** | **0.073452884** |
| **23.55142** | **0** | **0** | **391.2273** | **245.9143** | **0.936789997** |
| **1.358736** | **0** | **0** | **358.625** | **296.7931** | **0.911564803** |
| **1.59917** | **12.5** | **12.5** | **17214** | **344.28** | **0.911564803** |
| **3.140189** | **0.5** | **0.5** | **3442.8** | **344.28** | **0.595401972** |
| **6.476639** | **0.073964** | **0.073964** | **662.0769** | **358.625** | **0.677649082** |
| **2.564065** | **0.198217** | **0.198217** | **637.5556** | **358.625** | **1.43E-04** |
| **6.476639** | **2.42** | **2.42** | **1721.4** | **358.625** | **0.902433272** |
| **72.61686** | **15.68** | **15.68** | **3442.8** | **358.625** | **0.665728529** |
| **24.94034** | **1.300728** | **1.300728** | **555.2903** | **358.625** | **0.825561533** |

Table 2: Reliability Model Values

## 6 PERFORMANCE MEASURES

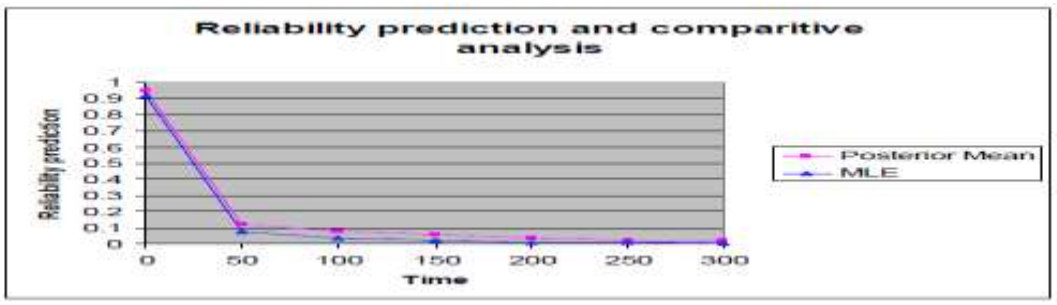

Figure 7: Reliability prediction and comparative Analysis.

In general, the reliability of the system decreases as time increases but the reliability of posterior mean always lies above the mean of MLE. Both point estimate methods of MLE and posterior mean can predict a close reliability trend, so the new method using the posterior mean can be an alternative way for the point estimate of the parameters. More importantly, using the posterior probability distribution, we can further analyze the uncertainty of the predicted reliability. In addition, the confidence intervals depend on both the prior distribution and the new observations.

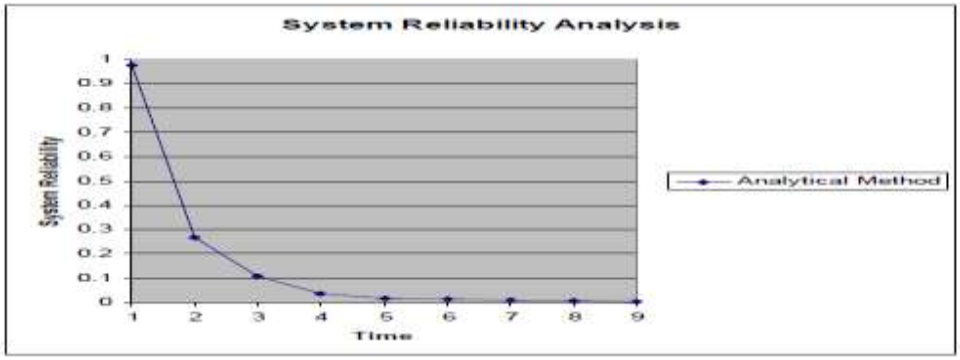


Figure-8: System Reliability Analysis

From the analytical results of uncertainty analysis in Fig. 8, we find that, during the initial period of the reliability prediction, the system reliability is high, indicating that the uncertainty of the software reliability is low. Then, the confidence interval increases and reaches the maximum around the middle part. At the latter part, the mean value of system reliability is also small so that the comparative uncertainty is still large.

## 6. CONCLUSIONS

We have described an approach to the probabilistic modeling and analysis of software systems. In This proposed work gives the solution for uncertainty problems in reliability modeling on system level. This safe growth model solves the challenges for the dearth of data by using quality factors. From the similar models, we have taken expert knowledge, historical data, and developmental environments .This expert knowledge involved in analyzing the uncertainty and for compensating insufficient failure data. After analyzing the problem, this work further extends to more complicated systems that contain numerous components, each with its own respective distributions and uncertain parameters. The model that we have described exploits the expert judgments of the tester. If these judgments are overly simplistic, then the model will not give a good representation of the faults in the software. In such cases, the model will still improve on the analysis of the tester, but with the additional advantage that the various assumptions of the tester may be explicitly scrutinized within the model by the model diagnostics for the BGM. These diagnostics are based on discrepancies between the

actual test behavior and the predicted behavior according to the model; for example, the prediction of few faults in a given subarea might be contradicted by observation of several faults in that area. Finally, note that the BGM approach captures, as far as is deemed practicable, the expertise of the tester and maintains it in usable form as a knowledge base. This represents a considerable resource to the software owner, who is routinely faced by the problem of the expertise of a tester being lost due to everyday practicalities such as personnel changes. The relevance of the BGM approach to support software testing, from a management perspective, is addressed inwhich also considers the circumstances under which this approach has been developed. In a future paper, we shall address how we may assess the viability of the approach in terms of the scale and complexity of the software to be tested.

## REFERENCES

M. Neil and N. Fenton, "Predicting Software Quality Using Bayesian Belief Networks," Proc. 21st Ann. Softwar Eng. Workshop NASA/Goddard Space Flight Center. 1996.

U. Kjirulff and L.C. van der Gaag, "Making Sensitivity Analysis Computationally Efficient," Proc. 16th Conf. Uncertainty in Artificial Intelligence. 2000.

G. Rothermel and M. J. Harrold, "Analyzing Regression Test Selection Techniques," IEEE Trans. Software Eng. vol. 22, pp. 529551, 1996.

M. Bouissou, F. Martin, and A. Ourghanlian, "Assessment of a SafetyCritical System Including Software: A Bayesian Belief Network for Evidence Sources," Proc. Ann. Reliability and Maintainability Symp.. RAMS '99. 1999.

M. Ortega, M. Perez, & T. Rojas, "Construction of systemic quality model for evaluating a software product", Software Quality Journal 11: 219-242, 2003.

E.T. Jaynes, "Information Theory and Statistical Mechanics," Statistical Physics, pp. 181-218, 1963. C.Y. Tseng, "Entropic Criterion for Model Selection," Physica A: Statistical and Theoretical Physics, vol. 370, no. 2, pp. 530-538, 2005.

K.S. Trivedi, Probability and Statistics with Reliability, Queuing, and Computer Applications. Prentice-Hall, 1982.

D.A. Wooff, M. Goldstein, and F.P.A.Coolen,"Bayesian Graphical Models for Software Testing," IEEE Trans. Software Eng., vol. 28, no. 5, pp. 510-525, May 2002. IEEE. "IEEE standard for a software quality Metrics Methodology", 1993.

M. Xie, Y.S. Dai, and K.L. Poh, Computing System Reliability: Models and Analysis. Kluwer Academic, 2004.

Liao H,Enke D., and Wiebe H.,"An Expert Advisory Systems for ISO 9001 Quality System",Expert Systems with Applications,Vol 27,pp.313-322,2004.

J. Musa, A. Iannino, and K. Okumoto, Software engineering and managing software with reliability measures. :McGraw-Hill, 1987.

C.-Y. Huang and C.-T. Lin, "Software reliability analysis by considering fault dependency and debugging time lag," IEEE Trans. Reliability, vol. 53, no. 3, pp. 436–45 0, 2006.

P. Moranda, "Predictions of software reliability during debugging," in Proceedings of the Annual Reliability Maintainability Symposium, 1975.Vol. 02, No. 06, 2010.

P. Moranda and Z. Jelinski, Final Report on Software Reliability Study McDonnall Douglas Astronautics Company,1972, Tech. Rep.

J. Musa, "A theory of software reliability and its application," IEEE Trans. Software Engineering, pp. 312–327, 1975.

N. Karunanithi, D. Whitley, and Y. K. Malaiya, "Prediction of software reliability using connectionist models," IEEE Trans. Software Engineering, vol. 18, no. 7, pp. 563–574, July 1992.

G. A. Souza and S. R. Vergilio, "Modeling software reliability growth with artificial neural networks," in IEEE Latin American Test Workshop, Buenos Aires, Argentina, March 2006, pp. 165–170.

J. Holland, Adaptation in natural and artificial systems. : MIT Press, 1975.

E. O. Costa, S. R. Vergilio, A. Pozo, and G. A. Souza, "Modeling software reliability growth with genetic programming," in XVI International Symposium of Software Reliability Engineering, USA, November 2005, IEEE Computer Society.

G. Paris, D. Robiliard, and C. Fonlupt, "Applying boosting techniques to genetic programming," in IEEE International Joint Conference on Neural Networks, 2004, pp. 1163–1168.

D. Solomatine and D. Shrestha, "Adaboost-rt: A boosting algorithm for regression problems," Intelligent Artificial Evolution, pp. 312– 326, 2001.

R.M. Hierons and M.P. Wiper, "Estimation of Failure Rate Using Random and Partition Testing," Software Testing. Verification. and Reliability vol. 7, pp. 153-164, 1997.

G.L. Eyink and S. Kim, "A Maximum Entropy Method for Particle Filtering," J. Statistical Physics, vol. 123, no. 5, pp. 1071-1128, 2005.

A.L. Goel and K. Okumoto, "Time Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," IEEE Trans. Reliability, vol. 28, pp. 206-211, 1979.

W. Suryn, A. Abran, P. Bourque & C. Laporte, "Software product quality practices: Quality measurement and evaluation using TL9000 and ISO/IEC9126," Proceeding of the 10th International Workshop, Software Technology and Engineering Practice (STEP) 2002.

F.E. Norman & S.L. Pfleeger, Software Metrics: ARigorous and Practical Approach, Second Edition. Boston: PWS Publishing, 1997.

K. Khosravi, & Y.G. Gueheneuc, "A quality model for design patterns",2004.

M. Goldstein, "Subjective Bayesian Analysis: Principles and Practice," Bayesian Analysis, vol. 1, no. 3, pp. 403- 420, 2006.

B.R. Haverkort and A.M.H. Meeuwissen, "Sensitivity and Uncertainty Analysis of Markov-Reward Models," IEEE Trans. Reliability, vol. 44, no. 1, pp. 147-154, 1995.

D.E. Holmes, "Toward a Generalized Bayesian Network," Proc. Am. Inst. Physics Conf.Bayesian Inference and Maximum Entropy Methods in Science and Eng., vol. 872, pp. 195-202, 2006.

K. Rees, F.P.A. Coolen, M. Goldstein, and D.A. Wooff, "Managing the Uncertainties of Software Testing: A Bayesian Approach," Quality and Reliability Eng. Int'l. vol. 17, pp. 191203, 2001.

**Dejan Chandra Gope** is a student in the Department of Computer Science and Engineering at the University of Dhaka University of Engineering and Technology, Bangladesh. He received Bachelor of Science in Computer Science and Engineering (B.Sc) from the University of Dhaka University of Engineering and Technology in 2012. He is now studying Master of Science in Computer Science and Engineering (M.Sc) at the same University. His research interest lie in studying the role of computer vision and advanced human-computer interaction, Artificial Intelligence and Pattern Recognition.