Reliability of Component Based Systems – A Critical Survey

KIRTI TYAGI, ARUN SHARMA
Computer Science and Engineering, Computer Science and Engineering
Gautam Buddha Technical University,Lucknow
Sitapur road Lucknow
INDIA
Kirti.twins@gmail.com, arunsharma2303@gmail.com

Abstract: - Software reliability is defined as the probability of the failure free operation of a software system for a specified period of time in a specified environment. Day by day software applications are growing more complex and with more emphasis on reuse. Component Based Software (CBS) applications have emerged. The focus of this paper is to provide an overview for the state of the art of Component Based Systems reliability estimation. In this paper, we discussed various approaches in terms of their scope, model, methods, technique and validation scheme. This comparison provides insight into determining the direction of future CBS reliability research.

Key-Words: - Reliability, Failure, Markov Model, Component Based Systems, Software Architecture, Reliability Model, State Based Model, Path Based Model, Additive Model, Scenario, Operational Profile, Component Dependency Graph, Component, Failure Behavior, Non Homogeneous Poisson Process**,** Discrete Time Markov Chain.

1 Introduction

Software reliability theory is one of industry's seminal approaches for predicting the likelihood of software field failures. Software application reliability is defined as follows [1]:

- "The probability of a given system performing its task adequately for a specified period of time under the expected operating conditions".
- "The probability that software will provide failure-free operation in a fixed environment for a fixed interval of time".

Failure Probability is the probability that the software will fail on the next input selected. Software reliability is typically measured per some unit of time, whereas probability of failure is generally time independent. Software reliability differs considerably from program "correctness". A program is consistent with its specification, while reliability is related to the dynamic demands that are made upon the system and the ability to produce a satisfactory response to those demands. A correct program may be considered as unreliable, conversely a program that is not completely correct may be considered as reliable if the errors are insignificant or user can simply avoid the errors. Current Component Based Software Engineering (CBSE) is being popular among both researchers and practitioners due to its several advantages over object oriented approach. CBSE improves productivity, quality and reusability and reduce maintenance overheads and time to market.

Rest of the paper is organized as follows Section 2 gives the problems associated with software reliability. Section 3 describes the reliability models for CBS. Section 4 provides the common requirements for CBS reliability. Section 5 gives the proposed framework for characterization of different approaches. In Section 6, structural analysis of different approaches has been given. Paper is concluded with a summary and the description for future work in Section 7.

2 Problems with Software Reliability

The major difference between software and other engineering artefacts is that software is pure design. Software unreliability is always the result of design faults, which in turn arise from human failures. Hardware systems do fail through design and manufacturing defects more often than is desirable.

CBS reliability greatly depends upon the interaction among components. Interaction promotes dependencies. Higher dependency leads to a complex system, Hence reliability estimation will be difficult. Usually dependency is represented by an adjacency matrix. However, this representation

can check only for the presence of dependencies between components and does not consider the type of interactions. Interaction types have a significant contribution to the complexity of system, hence the reliability. Several reliability issues and metrics proposed by researchers for CBS. Sharma et.al. [2] propose a link list based dependency representation and implements it by using Hash Map in Java. This representation can store the dependency along with other information like, provided and required information can be used to analyze several interaction and dependency related issues.

3 Components Based Software Engineering

In Object-Oriented Programming (OOP) code is reused in the form of objects, and several mechanisms such as inheritance and polymorphism let the developers reuse these objects in several ways. The principle is the same with component-based software engineering (CBSE) also, but here the focus is on reusing whole software component, not just the objects.

CBSE comprises of two separate but related processes namely *component engineering* and *application engineering*. The former is concerned with the analysis of domains and development of generic and domain-specific reusable components while the latter involves application development using commercial off-the-shelf components (COTS) or components that have been developed in-house.

4 Advantages of CBSE

Developing software systems from existing components offer many advantages:

4.1 Flexibility

Run-time components can work independently, and, if designed properly, are much less dependent on their environment (hardware, system software, other applications or components).Therefore, component-based systems are much more adaptable and extendable than systems traditionally designed and built. Usually, components are not changed, but replaced. This flexibility is important in two areas:

4.1.1Hardware and system software:

Component based systems are less sensitive to changes in the foundation (for example: the operating system) than traditional systems. This results in a more rapid migration from one operating system to another or from one DBMS to another. An interesting result is also the possibility of a system in a technically heterogeneous environment.

4.1.2 Functionality

Component-based systems are at a functional level much more adaptable and extendable than traditional systems, because most of the new functionality can be reused some way or another or derived from already existing components.

4.2 Reusability

In principle, CBD enables the development of components which completely implement a technical solution or a business aspect. Such components can be used everywhere. functionality, Be it technical or business oriented, has to be developed and implemented just once, instead of several times, as is now typically the case. It will be clear this is a good thing from the point of view of maintainability, robustness and productivity. Of course, this reuse can be within a company, but also over several companies. This will be the case of components made by third-parties.

4.3 Maintainability

In a component-based system a piece of functionality ideally is implemented just once. It is self-evident this results in easier maintenance, which leads to lower cost, and a longer life for these systems. In fact, the distinction between maintenance and construction will become very vague, and completely disappear after some time. New applications will consist for a very large part of already existing components. Building a system will look more like assembly than really building. Moreover, the large, monolithic systems as we know them will disappear resulting in a blurring of the borders between the systems. It is also usual to mention the following points as advantages of CBD:

4.4 More rapid development/higher productivity of the developers

In principle, CBD will result in a more rapid development of systems, for reasons of reuse among others. In the long run, this higher productivity will be realized. However, in the short run the fruits of reuse will be smaller than the cost of the

introduction of a new way of system development. Furthermore, at his moment reusable components are not available in sufficient measure, and on top of that difficult to acquire.

5 Reliability Model for Component Based Systems

Computer system plays more and more important role in our daily lives as computer system failures can lead to a huge economic loss or even endanger human life. Reliability is one of the most important quality requirements of computer systems. A computer system comprises two major components, hardware and software. The growing importance of software dictates that the software reliability is the major stumbling block in highly dependable computer system. Researchers have focused on procedural and object oriented software reliability. However, at present, there is a lack of similar research effort for CBS. Furthermore, owing to certain specific features of CBS, existing reliability assessment frameworks for procedural or object oriented software cannot be applied as such to CBS. However, neither black box statistical software testing (BBSST) nor any other existing testing models are capable of adequately supporting modern CBS development techniques. To support these techniques, testing models are needed to:

1. Explain the dependency of failure probability for software on its components.
2. Exploit reused software components of known reliability in estimating overall software system reliability.

The above two points requires statistical models to describe the failure patterns for both individual software components and compositions of those components. The existing software reliability models for legacy systems are inappropriate for CBS. There is a need of such type of models which is based upon the system architecture. Many reliability models based upon the system architecture have been proposed. These models are known as Architecture Based Reliability Model. Architecture based software reliability techniques may be used for the following reasons:

- To develop a method that analyzes the application reliability built from the COTS software components.
- To understand system reliability dependency on individual component reliabilities and their interconnection mechanism.

Swapna S. Gokhale [3] proposed some limitations for architecture based analysis technique. She classified the limitations into five categories namely modelling, analysis, parameter estimation, validation and optimization. Modelling limitations include concurrent execution, non markov transfer of control, non exponential sojourn time, and interface failures etc. Analysis limitation includes reliability estimation, sensitivity and interface analysis, uncertainty quantification etc.

4 Common Requirements for Architecture Based Reliability Models

5.1 To Identify the Component

Standard software engineering concept of a component is the basic entity in the architecture based approach. A component is conceived as a logically independent entity of the system which performs a particular function. Component can be independently designed, implemented, and tested. User can define the component which depends on the factors such that probability of getting required data.

5.2 Software Architecture

Software architecture is the way of defining the software behaviour with respect to the manner in which different software components interact with each other.

5.3 Failure Behavior

Failure behaviour is also associated with software architecture. Components failure behaviour can be expressed in terms of their reliabilities or failure rates.

5.4 Combining the architecture with the failure behavior

There are three different approaches that are used to combine the software architecture with the failure behaviour. These approaches are namely: state based approach, path based approach and additive approach.

5.4.1 State based models

5.4.1.1 Definition

In this model the control flow between components has been taken into consideration. It is assumed that

components failed independently. In these models transfer between components has been considered as Markov behaviour, which means that current components behaviour is independent of past behaviour. On the basis this model, architectural model using Discrete Time Markov Chain (DTMC) or semi Markov Processes have been proposed. These can be represented by two methods:
Hierarchical model: In this model we first solve architecture model and then superimpose the failure behaviour on the solution of the architecture model in order to predict reliability.

Composite model: This model combines the software architecture with the failure behaviour into a composite model which is then solved to predict application reliability.

5.4.1.2 Failure Behavior

It can be represented by three types:

1. Failure Probability or Reliability.
2. Constant Failure rate.
3. Time Dependent Failure Rate
4.

5.4.1.3 Limitations

Failure probability cannot be constant; more time spent in a component, the higher the constant failure rate as the failure model can account for this fact.

5.4.2 Path based models

5.4.2.1 Definition

These models consider the possible execution paths for estimating the application reliability. Sequence of components along different paths is obtained by testing either experimentally or algorithmically.
Reliability of each path = product of reliabilities of components along that path.
System Reliability = average of path reliabilities over all paths

5.4.2.2 Failure Behavior

Failure probability is used to represent failure behaviour.

5.4.2.3 Limitations

These models provide only an approximate estimate for application reliability when the application architecture has infinite paths due to the presence of loops.

5.4.3 Additive models

5.4.3.1 Definition

These models focus on estimating system reliability based on the components failure data.

5.4.3.2 Failure Behavior

Component failure can be modelled as a non homogeneous Poisson process. Additive models do not explicitly consider software architecture.

5.4.3.3 Limitations

These models consider software reliability growth. Additive models do not explicitly consider the software architecture.

6 Proposed Framework for Characterization of Different Approaches

Proposed framework for reviewing the software reliability research is based upon following factors:

6.1 Scope

Scope is the software applications domain on which the proposed approach is applicable. Basically the approach for following types of applications has been reviewed:

- Component Based Systems
- Complex Component Based Systems
- Service Oriented Architecture (SOA)
- Reuse Oriented Systems.

CBS has been defined in Section 2. Complex component based systems are used for very large CBS applications. Reuse oriented systems are the systems development in which a program is refined by producing a sequence of prototypes called models. Each of these is automatically derived from the preceding one according to a sequence of defined rules. A SOA is a collection of web services. These services can communicate with each other. During communication services simply pass the data or it involves two or more services to perform some activity.

6.2 Model

There are three main models on which the reliability analysis approaches are based

- State based Models.
- Path based Models
- Additive Models.

The definition of these models is given in Section 5

.

6.3 Technique

In the proposed approach reliability may be estimated by following types:

- Using mathematical formulas
- Using Component Dependency Graph (CDG)
- Using Algebraic Method
- Algorithm

6.4 Validation

We have selected following parameters to check whether the proposed approach is validated or not:

- No validation
- Validated through few experiments
- Fully validated

6.5 Critique

This is the assessment of noticeable features of the proposal.

7. Structural Analysis for Reliability Model

7.1 Littlewood's model for reliability estimation [4]

Scope

COTS Components Based Software Systems.

Model

State Based Model.

Method

Software architecture can be described by an irreducible SMP thus generalizing the previous work which describes software architecture with Continuous Time Markov Chain (CTMC). The program comprises a finite number of modules and the transfer of control between modules is described by the probability p_{ij} = P*r* {program transit from module *i* to module *j*}. The time spent in each module has a general distribution F_{ij} (t) with a mean sojourn time μ_{ij}.

Failure behavior:

1. During their execution individual components fail with constant failure rates.
2. The interface between components can also be fail.

Technique

CTMC is used to represent whole application reliability.

Validation

Validation is done using a hypothetical example.

Critique

This model is an earliest; still it is a general architecture based software reliability model.

7.2 Cheung's model for reliability estimation [5]

Scope

COTS Components Based Software Systems.

Model

State Based Model.

Method

This model considers software reliability in two terms:

1. Component utilization
2. Component reliabilities.

It is assumed that the program flow graph has a single entry and a single exit node and that the transfer of control among modules can be described by DTMC with a transition probability matrix P = $[p_{ij}]$.

Technique

The Solution for this method is taken as composite.

Validation

Validation is given based on some example.

Critique

This model indicates that program components used most often during execution time probably are the critical module from a reliability point of view.

7.3 Kubat's Model for reliability estimation [6]

Scope

COTS Components Based Software Systems.

Model

State Based Model.

Method

This model includes the information about component execution time, this result in an SMP as a model of software architecture. In this model it is assumed that the transition among the component follows DTMC such that:

$Q_i(r)$ = probability that task *r* will first call component *i*

P_{ij} (r)= probability that task *r* will call component *j* after the execution of component *i*

Failure intensity for a component *i* is given by α_i.

Technique

The solution of this method is taken as hierarchical.

Validation

Validation is given by a hypothetical example.

Critique

This is a good model for reliability estimation. This model considers the case for software composed of M components designed for K different tasks.

7.4 Gokhale's Model for reliability estimation [7]

Scope
COTS Components Based Software Systems.
Model
State Based Model.
Method
This model considers the time dependent failure rates and utilization of the components through the cumulative expected time spent in the component per execution. In this method application description is given by an absorbing DTMC. Testing is used to determine application architecture. A coverage analysis tool ATAC is used to determine the branching probabilities between the components. Component Failure Behavior is given by a NHPP (Non Homogeneous Poisson Process).
Technique
The solution of this method is taken as hierarchical. The Symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE) that solves stochastic models for reliability, performance, and performs ability were selected.
Validation
The application5 is tested using the minimal test set with 40 random orderings of the test cases for each file and the coverage was measured for each run.
Critique

This approach describes an analytical model to reliability estimation. Parameters for this model determined experimentally.

7.5 Everett's model for reliability estimation [8]

Scope
COTS Components Based Software Systems.
Model
Additive Model
Method
In this approach, Component's reliability is analyzed using the Extended Execution Time (EET) model. Parameters for this model can be determined from software properties.
Technique
This model used NHPP.
Validation
Validation is done using some example.
Critique
This is an earliest and good model for analyzing CBS reliability. In this approach it is required to keep track of cumulative amount of processing time that is spent in each component.

7.6 H. Singh's Bayesian Approach to Reliability Prediction [9]

Scope
Components based and reuse oriented software development paradigms.
Model
Path Based model.
Method
In this approach Unified Modelling Language (UML) technique is incorporated into reliability prediction and assessment. The technique for reliability analysis at the design level i.e. before system development and integration level is given.
Technique
Algorithmic and probabilistic technique is used based on UML diagrams.
Validation
Validation is given through case study and it is given that reliability estimated by this approach is highly accurate.
Critique

This is a new approach for reliability assessment of CBS. The proposed approach is based on UML diagram hence the reliability can be predicted in early design phase. This approach is scalable because all the calculations are done by an automated tool. There is one limitation for this approach that if new scenarios are taken into account that reliability prediction algorithm is taken as new and separate operational profiles are generated.

7.7 Sherif Yacoub's Scenario Based Reliability Analysis Approach [10]

Scope
COTS Components Based Software Systems.
Model
Path Based Model.
Method
Scenario-Based Reliability Analysis Approach (SBRA) is based on usage scenarios. The execution profile for these scenarios is assumed to be available. CDG is extended for complex distributed systems by incorporating the effect of component distribution and hierarchy of sub systems. A stack based algorithm is discussed to analyze the system level reliability sensitivity as a function of component and subsystem reliabilities. The same algorithm is applicable to analyze the sensitivity of subsystem reliability as a function of component reliabilities and link reliabilities.
Technique
An algorithm is proposed to analyze the reliability that is based upon scenarios. Various types of

probabilities are used to calculate the constraints of algorithm. The constraints are calculated using some mathematical formula.

Validation

Validation is done on waiting queue simulator. This is a real world example and it is seen from the validation that overall application reliability is directly proportional to the few components reliabilities which are in frequent use.

Critique

The proposed approach is good it can be extended for distributed systems also but this approach has some limitations also:

1. Failure dependencies among components are not considered.
2. This approach is based on the static analysis of execution scenario.

This model does not consider the dynamic for the execution of the application. It is applicable for sensitivity analysis of application reliability to component, subsystem, and interface reliability given the average scenario execution time.

7.8 Fan Zhang's novel model for CBS reliability analysis [11]

Scope

COTS Components Based Software Systems.

Model

Path Based Model.

Method

This approach is based on a sub domain based method which is Path based architectural reliability model, and provides the enhanced composition algorithms to solve the model.

Technique

The model is based upon the CDG. In this approach a systems operational profile is given. It is assumed that control flow transits from component *i* to component *j*, component *j* reliability is calculated as Tij . ($R_{ij} \times W_{ij}$).

T_{ij} = the transition probability from component *i* to component *j*

R_j = reliability vector on each sub domain of component *j*

W_{ij} = the weight vector for each sub domain of component *j* in transition from component *i* to *j*.

Validation

Hypothetical validation is given .There is no real time validation. This approach is able to characterize application reliability on change in the operational profile. It is given that the proposed model is able to analyze the components reliability when the systems operational profile changes.

Critique

The proposed approach analyzes the CBS reliability and provides the algorithm which is based on component's composition to estimate the reliability. Effect of different operational profiles is seen by using this model up to a great extent.

7.9 Ning Huang's An Algebra-Based Reliability Prediction Approach [12]

Scope

Components based software systems and composite web services.

Model

Architecture and operational profile based approach.

Method

The approach is based on algebra. This gives a framework for describing the syntax and predicting the reliability that is implemented on Maude.

Technique

Algebraic method is given.

Validation

A hypothetical example is considered.

7.10 WANG Dong's Reliability Analysis of CBS on Relationships of Components [13]

Scope

COTS Components Based Software Systems and SOA.

Model

Path Based model.

Method

Various complex components relationships are analyzed. These complicated relationships are solved using Markov model. These relationships have great impact on systems reliability .On the basis of these results a new tool has been developed to calculate the software application reliability.

Technique

Mathematical formulas are used to calculate relationship among components.

Validation

No validation is given.

Critique

Markov model scope has been extended by using this mechanism. But this approach has some limitations. It assumes that all components reliabilities and transition probabilities are given. However, in practice, this is not always true.

7.11 Vivek Goswamis's Method for reliability estimation [14]

Scope

COTS Components Based Software Systems.

Model

Component usage ratio based.

Method
This approach is relatively simple and implementable. It estimates overall system reliability on the basis of component usage ratio and individual component reliability. In this approach with the help of operational profile component usage ratio is taken into consideration.
Technique
Components reliability is calculated using some mathematical formulas.
Validation
No validation is given.
Critique
Component usage ratio can be flexible so that given approach may be used in real time applications.

7.12 Yuanjie Si's Reliability Estimation Framework through Component Composition Mechanisms [15]

Scope
COTS Components Based Software Systems.
Model
Path Based Model.
Method
In this approach, five basic component composition mechanisms and their reliability estimation techniques are proposed. After calculating the reliability for each composition, a procedure is given to estimate the overall application reliability.
Technique
Mathematical formulas are used to calculate reliability for each composition then procedural approach is used.
Validation
Validation is given through case study.
Critique
The proposed approach estimates the reliability based on component composition mechanism and component utilization frequency. The accuracy is also good. More composition mechanism may be recognized.

7.13 Chao-Jung Hsu's Adaptive Reliability Analysis Using Path Testing [16]

Scope
Complex Components Based Software Systems.
Model
Path Based model.
Method
This is an adaptive approach for testing path into reliability estimation for complex component based systems. For path reliability estimation three methods have been proposed namely sequence, branch and loop structures. The proposed path reliability can be used for reliability estimation of overall application.
Technique
Algorithmic approach is used.
Validation
Experimental results are given for showing the path reliability and correlation coefficient with actual reliability is calculated.
Critique
A promising estimation of software reliability can be given by this approach when testing information is available. A sensitivity analysis is also performed to know the effect of each node on the system reliability.

8 Conclusion and Future Work

In this paper, various available reliability analysis models for component based software applications are examined. We considered some criteria on basis of which we examined the available approaches as scope, model, technique, method and critique. Most of the proposed approaches are mathematical and based upon the operational profile. To calculate the overall application reliability existing work take two important considerations one is reliability of individual component and another is operational profiles of the system. However, soft computing techniques may produce better results. For reliability there is still a good scope to estimate it using soft computing techniques.

References :

[1] ANSI/IEEE, 1991 Standard Glossary of Software Engineering Terminology, *STD-729-1991,*
[2] Arun Sharma, P.S.Grover, Rajesh Kumar, 2009, Dependency Analysis for Component Based Software Systems, ACMSIGSOFT Software Engineering Notes Vol. 34, No 4 pp 1-6.
[3] S. Gokhale et al.2007, Architecture Based Software Reliability Analysis: Overview and Limitations. IEEE Transactions on Dependable and Secure Computing. pp 32-40.
[4] B. Littlewood. 1975 A reliability model for systems with markov structure. Applied Statistics, 24(2):172–177.
[5] R. Cheung, 1980 A user-oriented software reliability model. IEEE Trans. on Software Engineering, 6(2):118–125.
[6] P. Kubat, 1989 Assessing reliability of modular software.*Operations Research Letters*, 8:35–41.
[7] S. Gokhale *et al.*,1998 Reliability simulation of Component based software systems, in *Proceeding of. 9th International. Symposium on Software*

Reliability Engineering (ISSRE Nov 98), Paderborn, Germany, pp. 192–201.

[8] W.Everett , 1999 Software Component Reliability Analysis, in *Proceeding of.* Symposium *Application Specific systems and Software Engineering Technology* (ASSET'99) ,pp 204-211.

[9] H. Singh, V. Cortellessa,B. Cukic,E. Gunel and V. Bharadwaj ,2001, A Bayesian approach to reliability prediction and assessment of component based systems, in *12th IEEE International. Symposium on Software Reliability Engineering* , (Hong Kong, Nov. 2001), pp. 12–21.

[10] S. Yacoub, B. Cukic, and H. Ammar.2004, A scenario-based reliability analysis approach for Component based software .IEEE Transactions on Reliability, 53(4):465–480.

[11] Fan Zhang , Xingshe Zhou, Junwen Chen, Yunwei Dong,2008 , A Novel Model for Component –based Software Reliability Analysis, *11th IEEE High Assurance Systems Engineering Symposium* , pp 303-309.

[12] Ning Huang ,Dong Wang, Xiaoguang JIA,2008, FAST ABSTRACT: An Algebra – Based Reliability Prediction Approach for Composite Web Services,*19th International Symposium on Software Relability Engineering* , pp 285-286.

[13] Wang Dong , Ning Huang, Ye Ming,2008, Reliability Analysis of Component –based Software Based on Relationships of Components, *IEEE Conference on Web Services* pp 814-815 .

[14] Vivek Goswami, Y.B.Acharya,2009, Method for Reliability Estimation of COTS Components based Software Systems *International. Symposium on Software Reliability Engineering (ISSRE 2009)*.

[15] Yuanjie Si, Xiaohu Yang, Xinyu Wang,Chao Huang,Aleksander J. Kavs, 2011, An Architecture-Based Reliability Estimation framework Through Component Composition Mechanisms, *2nd International Conference on Computer Engineering and Technology* , pp 165-170 .

[16] Chao-Jung Hsu and Chin-Yu Huang, 2011, An Adaptive Reliability Analysis Using Path Testing for Complex Component based Software Systems, IEEE transaction on reliability Vol. 60, No 1 pp 158-170.

[17] Capers Jones. "Software Estimating Rules of Thumb" available at ieeexplore.ieee.org/iel1/2/10412/00485905.pdf?renumber=485905

[18] Hsinchun Chen. Machine "Learning for Information Retrieval: Neural Networks, Symbolic Learning, and Genetic Algorithms" available at http://ai.bpa.arizona.edu/papers/mlir93/mlir93.html

[19] Richard E. Fairley "Recent Advances in Software Estimation Techniques" Management Associates Woodland Park, CO, USA 1992.

[20] I Sommerville. "Software Engineering", Sixth Edition". Addison-Wesley Publishers Limited, 2001.

[21] Construx Software Inc. available at http://www.construx.com/estimate

[22]International Software Benchmarking Standards Group (ISBSG), "Software Estimation

[23]Martin Shepperd, Chris Schofield, Barbara Kitchenham "Effort Estimation Using Analogy" IEEE Proceedings of ICSE, 1996.

[24]Kitchenham B (1997) Counterpoint: the problem with function points. IEEE Softw 14(2):29.

[25]Bohem, B , Software Engineering Economics, Prentice-hall, Inc.,1981

[26] J . Smith: "the estimation of Effort Based on use Cases", rational Software whit paper,(1999).

[27] derger, J., Function Point Analysis, Prentice-Hall,Inc .,1989.

[28] Gennaro Costagliola and Genoveffa Tortora, "Class Point: An Approach for the Size Estimation of Object- Oriented Systems," IEEE Transactions on Software Engineering, Vol. 31, No. 1, pp. 52-74, Jan. 2005.(6)

[29] W. Farr. Handbook of "Software Reliability Engineering"(1996), chapter Software Reliability Modeling Survey, pages 71–117. McGraw-Hill, New York, NY.

[30] S. Gokhale and M. R. Lyu.(1997) "Regression Tree Modeling for the Prediction of Software Quality". In Proc. of ISSAT '97, pages 31–36, Anaheim, CA.

[31] S. Gokhale, T. Philip, P. N.Marinos, and K. S. Trivedi. (1996)"Unification of Finite Failure Non-homogeneous Poisson Process Model through Test Coverage". In Proc. Intl. Sypmosium on Software Reliability Engineering (ISSRE '96), pages 289–299, White Plains, NY.

[32] D. Hamlet. (1992)"Are We Testing for True Reliability ?".IEEE Software, 13(4):21–27.

[33] M. Hecht, D. Tang, H. Hecht, and R. W. Brill.(1997) "Quantitative Reliability and Availability Assessment for Critical Systems Including Software". In Proc. of Computer Assurance '97, pages 147–158, Gatheirsburg,Maryland.

[34] J. R. Horgan and S. London.(1992) "ATAC: A Data Flow Coverage Testing Tool for C". In Proc. of Second Symposium on Assessment of Quality

Software Development Tools, pages 2–10, New Orleans, Louisiana.

[35] J. R. Horgan and S. A. London.(1991) "Dataflow Coverage and the C Language". In Proc. of the Fourth Annual Symposium on Testing, Analysis and Verification, pages 87–97, Victoria, British Columbia, Canada.

[36] J. R. Horgan and A. P. Mathur. (1996) Handbook of SoftwareReliability Engineering, M. R. Lyu, Editor, chapter Software Testing and Reliability, pages 531–566. McGraw-Hill, New York,.

[37] T. M. Khoshgoftaar, B. B. Bhattacharyya, and G. D. Richardson. (1992)"Predicting Software Errors, During Development, Using Nonlinear Regression Models: A Comparative Study". IEEE Trans. on Reliability,41(3):390–395.

[38] M. Lipow.(1982) "Number of Faults per Line of Code". IEEE Trans. on Software Engineering, SE-8(4):437–439.

[39] D. L. Parnas. (1975) "The Influence of Software Structure on Reliability". In Proc. 1975 Int'l Conf. Reliable Software,pages 358–362, Los Angeles, CA.

[40] R. A. Sahner, K. S. Trivedi, and A. Puliafito(1996). "Performance and Reliability Analysis of Computer Systems:An Example-Based Approach Using the SHARPE Software Package". Kluwer Academic Publishers, Boston.