COMPONENT-BASED HETEROGENEOUS SOFTWARE ARCHITECTURE RELIABILITY (COHAR) MODELING

S. Ramamoorthy# Dr. S. P. Rajagopalan* S. Sathyalakshmi@

ABSTRACT:

In this paper, we propose an analytical model for component-based heterogeneous software architecture reliability and a method to find the solution for finding the optimal reliability of the overall software system according to the reliability of each component, the operational profile, and the architecture of software. Our approach is based on Markov chain properties and architecture perspectives to state view transformation in order to compute the reliability on heterogeneous software architecture consisting of various styles.

Key Words: *Heterogeneous architecture, Markov chain, Transition Matrix, Software reliability*

1. INTRODUCTION:

Software reliability is one of the key metrics for determining the quality of software. It is often defined as the probability of a failure-free operation of a computer program within a specified exposure time interval. Most of the analytical models, developed for measuring reliability, focus on observing the behavior of software, based on an operational profile and not on software architecture. Software architecture is defined as the structure of software at an abstract level, consists of a set of components, connectors and configurations. Modern software often embodies complex heterogeneous architecture to achieve multiple quality requirements, such as the use of a parallel architecture to increase performance and/or introduce a back-up component to provide fault tolerance. Recent research efforts have been focused on the development of approaches to predict the reliability of a software application taking into account its architecture.

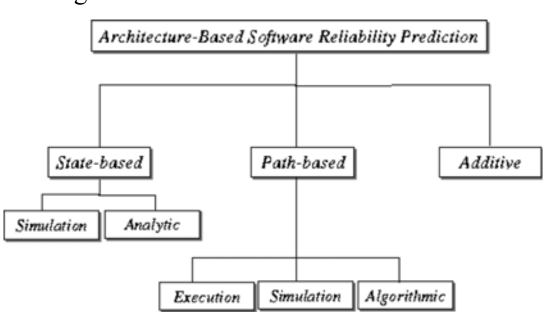


Fig 1. Classification of architecture-based software reliability models

2. COMPONENT-BASED RELIABILITY:

Goseva-Popstojanova *et al.* classify the existing architecture-based models into three broad categories: state-based, path-based, and additive. State-based models use the control graph to represent software architecture, and predict reliability analytically. Path-based models compute software reliability considering the possible execution paths of the program. The execution paths may be determined using simulation, by executing the application, or algorithmically. Additive models assume that each component reliability can be modeled by a non-homogeneous Poisson Process (NHPP), which leads the system failure process to be NHPP with cumulative number of failures & failure intensity functions that are the sums of the corresponding functions for each component. Additive models do not consider the architecture of the application explicitly. The broad classification of architecture-based software reliability models is shown in Fig. 1.

The state based model can be thought of as follows. The state diagram is usually used to depict the system behavior. The node Si represents system state i and the transition from state Si to Sj is represented by a directed edge (Si, Sj) and an example of state diagram is given in Fig 2 below:

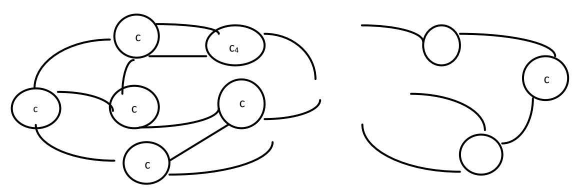


Fig. 2. The state diagram

The software architecture reliability model usually utilizes Markov chain to compute system reliability. Based on Markov chain properties, the transition between states is assumed as a Markov process. Let Ri denote the reliability of the component Ci, and Pij represents the probability of transition from component Ci to its successor component Cj . Based on this the transition matrix M (Fig3) is defined as given below, and the connector reliability is taken into account, so that Pij could be adjusted as the original transition probability multiplied by the reliability of the corresponding connector.

$$M = \begin{array}{c} \\ C1 \\ C2 \\ C3 \\ . \\ . \\ Ci \\ . \\ . \\ Cn\text{-}1 \\ Cn \end{array} \begin{array}{c} \begin{array}{cccccccc} C1 & C2 & C3 & \ldots\ldots & Ci & \ldots\ldots & Cn\text{-}1 & Cn \end{array} \\ \begin{bmatrix} 0 & R_1P_{12} & R_1P_{13} & \ldots\ldots & R_1P_{1i} & \ldots\ldots & R_1P_{1(n-1)} & R_1P_{1n} \\ R_2P_{21} & 0 & R_2P_{23} & \ldots\ldots & R_2P_{2i} & \ldots\ldots & R_2P_{2(n-1)} & R_2P_{2n} \\ R3P_{31} & R_3P_{32} & 0 & \ldots\ldots & R_3P_{3i} & \ldots\ldots & R_3P_{3(n-1)} & R_3P_{3n} \\ . & . & . & \ldots\ldots & . & \ldots\ldots & . & . \\ . & . & . & \ldots\ldots & . & \ldots\ldots & . & . \\ RiP_{i1} & R_iP_{i2} & R_iP_{i3} & \ldots\ldots & 0 & \ldots\ldots & R_iP_{ii(n-1)} & R_iP_{in} \\ . & . & . & \ldots\ldots & . & \ldots\ldots & . & . \\ . & . & . & \ldots\ldots & . & \ldots\ldots & . & . \\ R_{n-1}P_{(n-1)2} & . & . & \ldots\ldots & R_{n-1}P_{(n-1)i} & \ldots\ldots & 0 & R_{n-1}P_{(n-1)n} \\ R_nP_{n1} & R_nP_{n2} & R_nP_{n3} & \ldots\ldots & R_nP_{ni} & \ldots\ldots & R_nP_{n(n-1)} & 0 \end{bmatrix} \end{array}$$

Fig. 3. The Transition Matrix

3. RELATED WORK:

In his work, Roshanak Roshandel [17] discussed the uncertainty of the execution profile is modeled using stochastic processes with unknown parameters, the compositional approach to calculate overall reliability of the system as a function of the reliability of its constituent components and their (complex) interactions and sensitivity analysis to identify critical components and interactions will be provided. Lance Fiondella and Swapna S. Gokhale[18] considered the estimation of software reliability in the presence of architectural uncertainties and presented a methodology to estimate the confidence levels in the architectural parameters using limited testing or simulation data based on the theory of confidence intervals of the multinomial distribution. The sensitivity of the system reliability to uncertain architectural parameters was then quantified by varying the parameters within their confidence intervals. C. Smidts[19] presented an architecturally based software reliability model and underlines its benefits. The models based on an architecture derived from the requirements which captures both functional and non-functional requirements and on a generic classification of functions, attributes and failure modes. The model focuses on evaluation of failure mode probabilities and uses a Bayesian quantification frame work. Leslie Cheung and Leana Golubchik [22] discussed representative uncertainties which have identified at the level of a system's components, and illustrates how to represent them in the reliability modeling framework.

4. HETEROGENEOUS ARCHITECTURE RELIABILITY:

The main objective of this study is to compute the reliability of components-based heterogeneous software systems which may be comprised of various architectural styles. The architectural styles include sequential, parallel, fault tolerance and call-and-return styles. Most of the architectural styles can be viewed as the extension of these four basic styles and hence our study. In order to utilize the Markov model, a transformation for each architectural style from an architecture view to a state view is introduced. Based on the transformed view, the transition matrix M can be refined to obtain the style-based software reliability. The transition matrix M for various styles can be defined as follows:

1. Sequential Style: There are k components executed in a sequential order and there will be k states.

$$M(i, j) = Rj\ Pij \quad \text{when Si can reach Sj directly}$$
$$= 0 \quad \text{otherwise}$$

Where M(i, j) is the probability of successful transition of reaching state Sj from Si.

2. Parallel Style: Components are commonly running simultaneously and for k components, the transition matrix can be obtained as:

$$M(i, j) = Ri\ Pij, \quad \text{where Si not in Sp}$$
$$= \prod Rn\ Pnj, \text{ where } Cn \in Si, \text{ Si in Sp and for } 1 \le i, j \le |S| \;\&\; 1 \le n \le k$$
$$= 0, \text{ Si can not reach Sj}$$

In this case, the executions of the components C2 to Ck-1 are congregated into the state Spl which is an element of the parallel state set Sp. There are k components in which l = k-2 components are running concurrently into the same state; therefore, the total number of states is k-l+1. Because of the characteristics of parallel style, the transition probabilities from component C1 to components C2, C3, … and Ck-1 are all equal to P12, which is now the transition probability from state S1 to Spl. For convenience, we introduce {Si}, which returns the row number or column number of state variable Si in a matrix. Entry M({Spl}, {Sk}) requires that all the components from C2 to Ck-1 in state Spl perform successfully and finally reach Sk. Because the component reliabilities and transition probabilities are all independent of each other, the value of M({Spl}, {Sk}) is equal to ∏ Rn Pnj (where n varies from 2 to k-1) which is the product of all the component reliabilities in this state and the transition probabilities from components C2, C3, …., and Ck-1 to component Ck, respectively.

3. Fault Tolerance Style: This style consists of a primary component and a set of backup components, which may be used when one of the primary component fails. We assume that all backup components have the same probabilities as the primary components. Assuming K components in which l = k-4 components running as fault tolerance in the same state, the total number of states is k-l+1. The transition matrix can be constructed as follows:

M(i, j) =Ri Pij, where Si not in Sb

=Ra1+ Σ { [Π(1-Rm) , m=a1 to q-1] Rn, for q = a2 to ar} where Si in Sb and Si includes Ca1 to Car

= 0, where Si can not reach Sj; for 1 ≤ i, j ≤ |S| & 1 ≤ ar ≤ k

The transition probabilities from component C1 to components C2, C3,.. and Ck-3 are all equal to P12, which is now the transition probability from state S1to Sb1 (because concurrent characteristics of fault tolerance style is similar to parallel style). However, state S3 improves the reliability only when state S2 fails. Similarly, state S4 enhances reliability when both states S2 and S3 fail. Thus the reliability of reaching states Sk-1 and Sk-2 from S1, we have to consider when state S2 is always active, when state S2 fails but S3 is active, and so on. By induction, entry M(1, {Sbl}) is equal to R2 + Σ { [Π(1-Rm) , m=2 to n-1] Rn, for n = 3 to k-3}

4. Call-and-return Style: In this models, the execution of one component may request services from other components before transferring its complete control authority to others and like client-server style. Therefore, the called components may execute multiple times with only one time execution of the calling component. Assuming there are k components, the total number of states is therefore K. The transition matrix M can be constructed as follows:

M(i, j) = Ri Pij, where Si can reach Sj

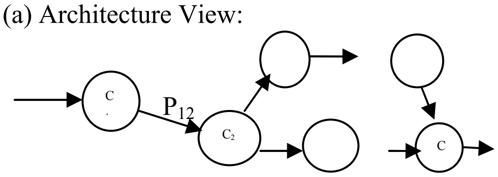
= Pij, where Si can reach Sj for 1 ≤ I, j ≤ k and Sj is a called component

= 0, where Si can not reach Sj

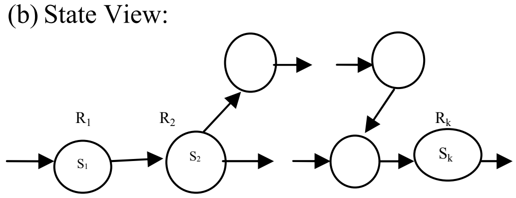
5. ARCHITECTURE AND STATE VIEW OF VARIOUS ARCHITECTURAL STYLES:

1. Sequential Style:

(a) Architecture View:

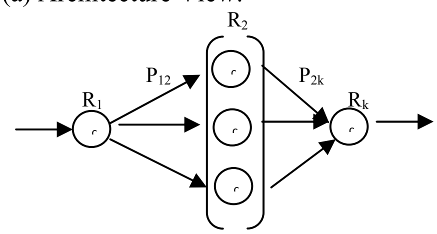


(b) State View:

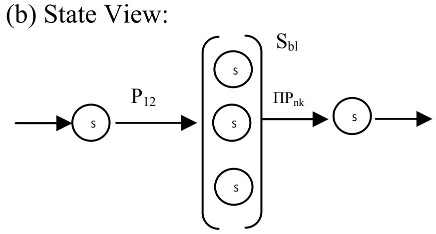


2. Parallel Style:

(a) Architecture View:

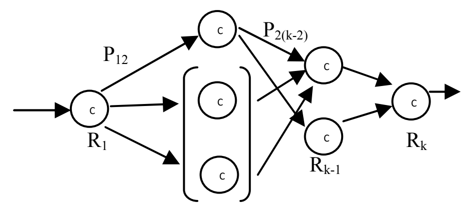


(b) State View:

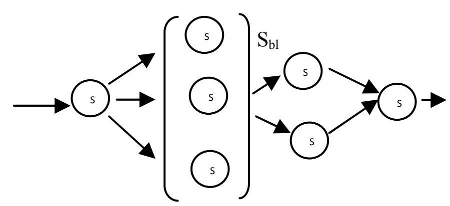


3. Fault Tolerance:

(a) Architecture View:

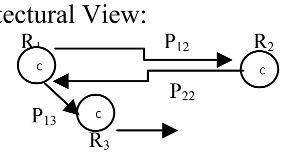


(b) State View:



4. **Call-and-Return Style:**

(a)Architectural View:



(b)State View:

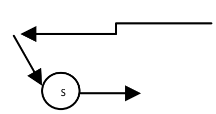


Fig 4: Various Architectural & the corresponding State views

6. THE RELIABILITY MODEL:

We have seen how to compute the transition matrix of a system based on a single architectural style in the previous section. For the heterogeneous styles, the transition matrix can be computed as shown in the following algorithm:

Input:

n –the number of components in the software

Pij - the probability of direct state transition from state Si to state Sj

Ri - the reliability of the component Ci

Case - Different architectural styles, say, 1for linear, 2 for parallel, 3 for fault tolerance and 4 for call-and-return styles

Output:

R –the overall reliability of the entire heterogeneous software system

Algorithm:

```
for i = 1 to n
    for j =1to n
        case = 1
            if state Si can reach state Sj directly then
                M(i, j) = Rj * Pij
            Else
                M(i, j) = 0
        end case 1
        case = 2
            if state Si is not in the group Sp then
                M(i, j) = Ri * Pij
            else if Ck is in Si, Si in Sp for I, j ≤ │S│ & 1 ≤ k ≤ n then
                M(i, j) = ∏ Rk Pkj
            else
                M(i, j) = 0
        end case 2
        case = 3
            if state Si is not in Sb then
                M(i, j) = Ri * Pij
            else if where Si in Sb and Si includes Ca1 to Car then
                M(i, j) = Ra1+ Σ { [ Π(1-Rm) , m=a1 to q-1] Rn, for q = a2 to ar}
            else if Si can not reach Sj for 1 ≤ i, j ≤ |S| & 1 ≤ ar ≤ k then
                M(i, j) = 0
        end case 3
        case = 4
            if state Si can reach state Sj then
                M(i, j) = Ri * Pij
            else if Si can reach Sj and Sj is a called component then
                M(i, j) = Pij
            else if Si can not reach Sj then
                M(i, j) = 0
        end case 4
    next j
next i
```

compute (the reliability of the overall system using)

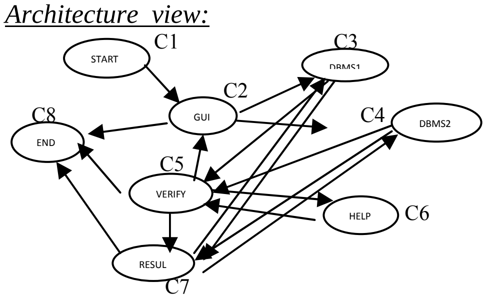
$$R = (-1)^{m+1} |E| / |I - M|$$

{where m is the number of columns / rows of the computed transitional matrix M in which all the fault tolerance components will be treated as a single component, |E| is the determinant value of the transition matrix M after deleting the first column and last row and |I - M| is the determinant value of the matrix (I – M).}

7. AN EXPERIMENT:

An example of on line examination is used to validate the correctness of the above reliability model. The fig.5 shows the architecture view and the corresponding state view of this system. The *Start* component is the initial component and the *End* component is the final component. Basically this system is working in sequential manner in addition to the following. Components DBMS1 and DBMS2 are categorized into fault tolerance style where DBMS2 is a backup for DBMS1. Components *Result* and *help* form a call-and-return style. Based on the architecture view and the information of style the matrix M is given below:

<u>*Architecture view:*</u>



<u>*State View:*</u>

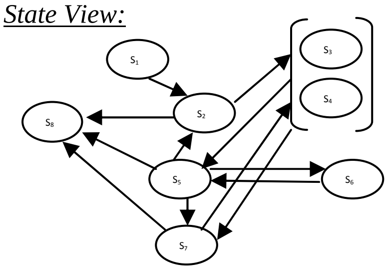


Fig 5: The architecture and state views of an online examination system

The reliability of each individual component, the transition probabilities between components, and the overall system reliability through experiment are given below:

The Reliability of components:
R1 = 1.0, R2 = 0.982, R3 = 0.97, R4 = 0.96,
R5 = 1.0, R6 = 0.996, R7 = 1.0, R8 = 0.99

The Transition Probabilities:
P1,2 = 1.0 P2,3 = P2,4 = 0.99 P2,8 = 0.001
P3.5 = 0.227 P4,5 = 0.227 P4,7 = 0.669
P5,2 = 0.048 P5,6 = 0.951 P5,7 = 0.104
P5,8 = 0.001 P6,5 = 1.0 P7,8 = 1.0

M =

	S1	S2	Sb1	S5	S6	S7	S8
S1	0	1.0	0	0	0	0	0
S2	0	0	0.981	0	0	0	0.001
Sb1	0	0	0	0.2267	0	0.662	0
S5	0	0.048	0	0	0.951	0.104	0.001
S6	0	0	0	1.0	0	0	0
S7	0	0	0	0	0	0	1.0
S8	0	0	0	0	0	0	0

Fig 6: The transition matrix

Here n = 7
Based on the concept of reliability and matrix theory, it was found that the Reliability of the overall system is given by :

$$T(1,n) = (-1)^{n+1} |E| / (|I - M|)$$

where |I - M| is the determinant of the matrix (I – M) and |E| is the determinant of the matrix excluding the first column and the last row of the matrix (I – M).
Thus the overall system reliability is

$$R = T(1, S8) = 0.559$$

8. CONCLUSION AND FUTURE WORK:

In this work, we demonstrated the working of **<u>CO</u>**mponent-based **H**eterogeneous **<u>A</u>**rchitectural **<u>R</u>**eliability (COHAR) Model and found that it works well within its scope. The future work shall be focused on:

(i) The sensitivity analysis on the reliability of the software architectural changes and
(ii) Finding the causes for improving the architectural reliability.

REFERENCES:

[1] Noel De Palma, Konstantin Popov, Nikos Parlavantzas, Per Brand, and Vladimir Vlassov - TOOLS FOR ARCHITECTURE BASED AUTONOMOUS SYSTEMS - Fifth International Conference on Autonomic and Autonomous Systems 2009

[2] Hamid Bagheri, Vajih Montaghami, Gholameraza SaFI, Seyed-Hassan Mirian-Hosseinbadi - AN EVALUATION METHOD FOR ASPECTUAL MODELING OF DISTRIBUTED SOFTWARE ARCHITECTURES - IEEE 2008

[3] Hans Van Vliet - SOFTWARE ARCHITECTURE KNOWLEDGE MANAGEMENT - 19th Australian Conference on Software Engineering 2008

[4] Odd Petter N. Slyngstad, Reidar Conradi, M.Ali Babar, Viktor Clerc, and Hans van Vliet - RISKS AND RISK MANAGEMENT IN SOFTWARE ARCHITECTURE EVOLUTION: AS INDUSTRIAL SURVEY - 15th Asia-Pasific Software Engineering Conference 2008

[5] Richard C. Holt - GROOKING SOFTWARE ARCHITECTURE - 15th Working Conference on Reverse Engineering 2008

[6] Xiao Xiao and Tadashi Dohi - ON EQUILIBRIUM DISTRIBUTION PROPERTIES IN SOFTWARE RELIABILITY MODELING - IEEE International Conference on Availability, Reliability and Security 2009

[7] Tirthankar Gayen - ANALYSIS AND PROPOSITION OF ERRRO-BASED MODEL TO PREDICT THE MINIMUM RELIABILTIY OF SOFTWARE - International Conference on Education Technology and Computer IEEE Computer Society 2009

[8] Tomotaka Ishii and Tadashi Dohi - A NEW PARADIGM FOR SOFTWARE RELIABILITY MODELING - FORM NHPP to NHGP - 14th IEEE Pacific Rim International Symposium on Dependable Computing 2008

[9] Bo Yang, Xiang Li - A STUDY ON SOFTWARE RELIBILITY PREDICTION BASED ON SUPPORT VECTOR MACHINES - IEEE Transactions on Reliability, Vol 57, No 1, 2008

[10] Hingguo Li, Xiaofeng Li and Yanhua Shu - AN EARLY PREDICTION METHOD OF SOFTWARE RELIABILITY BASED ON SUPPORT VECTOR MACHINE - IEEE 2007

[11] Zuzana KRAJCUSKOVA - SOFTWARE RELIABILITY MODELS - IEEE 2007

[12] Shiyi Xu - AN ACCURATE MODEL OF SOFTWARE RELIABILITY - 13th IEEE International Symposium on Pacific Rim Dependable Computing 2007

[13] Shiyi Xu - RECONSIDERATION OF SOFTWARE RELIABILITY MEASUREMENTS - 16th IEEE Asian Test Symposium 2007

[14] Shinji Inoue, and Shigeru yamada - GENERALIZED DISCERETE SOFTWARE RELIABILITY MODELING WITH EFFECT OF PROGRAM SIZE - IEEE Trasnactions on Systems, Man, and Cybernetics – Part A : Systems and Humans, Vol 37, No:2, 2007

[15] Alaa Sheta - PARAMETER ESTIMATION OF SOFTWARE RELIABILITY GROWTH MDOELS BY PARTICLE SWARM OPTIMIZATION - AIML Journal, Volume (7), Issue (1), 2007

[16] S. Chatterjee, S. S. Alam and R. B. Misra - SEQUENTIAL BAYESIAN TECHNIQUES : AN ALTERNATIVE APPROACH FOR SOFTWARE RELIABILITY ESTIMATION - Sadhana Vol 34, Part 2, 2009

[17] Roshanak Roshandel *Computer Science Department University of Southern California Los Angeles, CA 90089-0781 U.S.A. roshande@usc.edu* - CALCULATING ARCHITECTURAL RELIABILITY VIA MODELING **AND** ANALYSIS

[18] Lance Fiondella and Swapna S. Gokhale Dept. of Computer Science and Engineering Univ. of Connecticut, Storrs, CT 06269 {lfiondella,ssg}@engr.uconn.edu - SOFTWARE RELIABILITY WITH ARCHITECTURAL UNCERTAINTIES

[19] C. Smidts, University of Maryland, College Park, MD, 20704-753 1 ; csmidts@eng.umd.edu; D. Sova,

Intermetrics, Inc., 6301 Ivy Lane, Suite 200, Greenbelt, MD, 20770; dws@gblt.inmet.com; G. K. Mandela, University of Maryland, College **Park,** MD, 20704-753 1 ; gopi@eng.umd.edu AN ARCHITECTURAL MODEL FOR SOFTWARE RELIABILITY QUANTIFICATION

[20] Leslie Cheung, Roshanak Roshandel, Nenad Medvidovic, Leana Golubchik - EARLY PREDICTION OF SOFTWARE COMPONENT RELIABILITY

[21] Fan Zhang, Xingshe Zhou, Junwen Chen, Yunwei Dong *School of Computer Science and Engineering, Northwestern Polytechinical University, Xi'an, China{zhangfan, zhouxs, yunweidong, junwenchen}@nwpu.edu.cn* - A NOVEL MODEL FOR COMPONENT-BASED SOFTWARE RELIABILITY ANALYSIS

[22] Leslie Cheung, Leana Golubchik, Nenad Medvidovic, Gaurav Sukhatme {lccheung,leana,neno,gaurav}@usc.edu - IDENTIFYING AND ADDRESSING UNCERTAINTY IN ARCHITECTURE-LEVEL SOFTWARE RELIABILITY MODELING

Authors profile:

Author, S. Ramamoorthy, is a Professor in Computer Science and Engineering, Dr. MGR University, Chennai, Tamil Nadu, India. Has published 7 papers in National Conference and 1 paper in International Conference. This author can be contacted through srm24071959@yahoo.com

* Author, Dr. S. P. Rajagopalan, is a Professor Emeritus, Dr. MGR University, Chennai, Tamil Nadu, India. He has published more than 150 papers in reputed National & International Journals. Authored 6 books in computer science. Under his supervision 15 scholars obtained Ph.D degree in Computer Science from various Universities and can be contacted through sasirekaraj@yahoo.co.in

@ Author, S. Sathyalakshmi, is a Professor in Computer Science and Engineering, Hindustan University, Chennai, Tamil Nadu, India. She has published more than 10 papers in National Conference and 1 paper in International Conference. The author can be contacted through swamega@yahoo.com