Design of Simulator for Reliability Estimation of Component Based Software System

P K Suri¹, Sandeep Kumar²

¹ Professor, Department of Computer Science and Applications, Kurukshetra University, Kurukshetra (Haryana), India
² Assistant Professor, Department of Computer Applications, Dronacharya Institute of Management and Technology, Kurukshetra (Haryana), India.

Summary
This is the age of Rapid Application Development (RAD). To achieve the goal of RAD, Component Based Software Development (CBSD) can be of great help. Here Commercial-off-the-shelf (COTS) components are deployed together to make a larger application. These components are designed and developed independent of each other and independent of the application they are being used in. It has the advantages in the form of adaptability, scalability and reusability. Like in any other system here too the overall reliability of the system is a function of reliabilities of all the components used in the system and their interfaces with each other. In this paper an attempt has been made to compute the reliability of the system as a function of reliabilities of its components. Components along a path, called Course-of-execution, are executed during each simulation run. Starting from any component during any Course-of-execution, control is transferred to any other component as per the Markov process [7, 8, 9]. If we are able to reach the last component in the system, which is assumed to be a terminating component, that pass is assumed to be successful otherwise it is assumed to be unsuccessful. At the end we compute reliability of the system by dividing the number of successful passes by total no of simulation runs and get the reliability of the component based system.

Key words:
Component Based Software, CBSE, Simulation, COTS, Reliability

1. Introduction
According to Tausworthe [18] Reliability is one of the most important quality parameter of any system. It can be defined as probability that a system will perform as per requirements of the user for a specified period of time under given circumstances. According to [13] Customers want more reliable software faster and cheaper. Quantification of software reliability is very important. Juneja [7] and Shooman [14] have worked a great deal in comparing various component reliability models. Structure and architecture of software have a great impact on its correctness and reliability [14]. Component Based Software Engineering (CBSE) is the newest of the software development paradigms. In CBSE, idea is to compose, rather than develop, the software. Whenever some new application is to be developed, firstly market is searched for off-the-shelf components. If available, they are purchased and composed together, otherwise depending upon specification, components are developed in house. But there also main issue is to compose the components to make the application. Although Component Based technology has significantly reduced the development cost and time, quality control has become more difficult, since the system includes components from other systems [2]. Off-the-shelf components are commercially pre-tested and trusted [21], still it is very important to ensure the reliability of the software application, composed of these components. In conventional applications, system reliability can be estimated using system testing and system level architecture evaluation [6], but in case of Component Based applications, reliability can be estimated using the reliabilities of the individual components and their interfaces [21]. Most of the existing software reliability modeling techniques are black-box based where entire software is considered as single entity. They have the limitation of component testing information ignorance. Moreover they don’t take the software architecture into account. In some of the techniques available for reliability of Component based software, test cases are generated and faults are injected for studying the reliabilities of component based systems. But the problem with these techniques is that they can not be applied at early phases of the development life cycle. S. Gokhale, M Lyu and K Trivedi [3] used discrete event simulation to study the influence of various factors, individually and taken together, on various dependency parameters. In this paper they made use of two case studies. First one uses a terminating application in which effect of fault tolerance configurations of components on the failure behavior of the application have been studied, and second one uses a real time application with feedback control in which authors simulated the failure behavior of a single version considering reliability growth initially.

R. Michael [12] describes many simulation methods for software analysis and evaluation. Some of the techniques...
have been proposed in [4, 5]. But again all these are conventional methods and don’t take component based nature of a software into consideration.

S. Gokhale et al. [3] have measured system reliability from component reliabilities using an expression where system reliability is equal to product of component reliabilities (R = \prod_{k=1}^{n} R_k Where R_k s are component reliabilities). But component reliabilities are not binary in nature. According to [5] no component developer will ever guarantee the absolute correctness of a component. Each component has a non zero risk of failure called its unreliability and this unreliability in turn affects the overall reliability of the application. Reliability of that component is 1- failure probability.

J. Horgan et al. [6] used a UNIX utility as a component based system, which in turn may be a subsystem to some other system, to compute reliability of a component based system. The technique used has been named CBRE (Component Based Reliability Estimation). It uses sequence of components executed during system or subsystem testing. But at that time components were not independent entities, purchased off-the-shelf. They would rather be developed as modules of some application, exclusively developed for that application. But in case of CBSE, components are developed independent of an application and then deployed in different applications according to requirements.

Wang et al. [19] have given a reliability model for component based software systems where idea is to use the moving averages to compute the system reliability. The moving averages in the model provide an indicator that represents reliability growth movement within the evolution of a series of component enhancements. The model takes component configuration and reliability improvement as input and gives a series of reliabilities that are moving in average corresponding to discrete intervals, as output.

Wang et al. [20] have presented an analytical model for estimating the architecture based software reliability according to the architecture of the software application, reliability of each component, and their operational profile. Authors have performed analysis on heterogeneous software architecture styles like batch sequential, parallel, pipe filters, call and return and fault tolerant styles. Most of the models try to predict the reliability using observed failure data. Component based software is close to many real world systems, where any big system is made up off many small subsystems. As a matter of fact it is very difficult to obtain an analytical solution. So we propose this simulation based technique to compute the reliability of the application composed of reusable components. Suri and Aggarwal [16] have given a simple technique for evaluating reliability expression when logic flow of the program is governed by instructions in sequence, branch or parallel. However logic flow of an algorithm may be governed by a general network structure also [17]. Here an attempt has been made to implement this concept for the design of a simulator for reliability estimation of a Component Based Software System because in such a system transitions to various components along different courses of execution make a complex network.

2. Terms and Notations

The terms and notation for the simulator are given as under:

CCFG: Component Control Flow Graph

\{Ni\}: Set of i nodes in CCFG

\{Ej\}: Set of j Edges in CCFG

Interact (Ci,Cj) : Number of times C_i interacts with C_j during a course of execution.

TNC_{i,j} : Total number of interactions among all components during a Course-of-execution

PoTi\rightarrow{j}: Probability of Transition from component i to component j.

E: Number of courses of execution (paths)

PE_k: Probability of execution of path E_k

N: Total Number of Components

PoTi\rightarrow{j}: Probability of Transition from Component i to Component j

MPoT : “Probabilities of Transitions” Matrix

IPoT_{i\rightarrow{j}}: Imperfect “PoT” from i to j

MIPoT: Imperfect “PoT” Matrix

RoCi: Reliability of Component i

VRoC: “Reliabilities of Components” vector

STERM: Successful Termination

UTERM: Unsuccessful Termination

REL_{appl}: Overall reliability of the application
TRUNS: Total number of simulation runs

3. Simulator for the Component Based System

A simulation based model for computing the reliability of a component based system as, a function of component
reliabilities, is proposed here. The model being proposed here is based on the Markov chains and transition
probabilities. The flow of execution in the system is represented by a component Control Flow Graph (CCFG).
CCFG is a graph that consists of a set of nodes and edges
CCFG = \{<Ni, Ei>\}. Each node in the graph represents an
independent component. During a particular Course-of-
Execution (CoE), components along a path in the graph
are executed one by one, until the last node; called
Terminal Node (TERM) is reached. The transfer of control
along a path from one component to another component
takes place according to Markov process. The Markov
process states that if we are given the present state, the
future behavior of the system is independent of the past
behavior [11]. Each component has a specific reliability
associated with it, which is probability of successful
execution of that component. Once a component executes
successfully, it transfers control to one of the other
components depending upon the Course-of-execution.
This is a free flowing control where control can be
transferred from any component to any other component.
Probability of control being transferred to any component
from the current component is known as Transition
Probability. Probability of transition from one component
to another can be estimated using number of interactions
between two components using equation number 1 given
below [21]:

\[
PoT_i\rightarrow j = \sum_{k=1}^{N} PE_{k} \left( \frac{\text{Interact}(Ci,Cj)}{\text{Interact}(Ci,Cl) \mid l = 1,..N} \right)
\]  

In the control flow graph, each component corresponds to
a state of the Markov process. In every instance, the
execution starts with the first component and terminates
with the successful execution of the last component. In-
between, any number of transitions can take place among
various components. For example after successful
execution of the first component (which depends upon its
component reliability off course), transition may take
place to any other component (2, 3, 4,..., N) of the
system. Similarly if current state, or component, is 4,
transition may take place to any other component (1, 2, 3,
5,..., N) of the system. Transition will always depend
upon a specific course-of-execution. These transitions here
have been modeled using a “Probabilities of Transition”

matrix MPoT. Here PoT_\(i\rightarrow j\) is the probability that control
will be transferred to component ‘j’ provided, that the
component ‘i’ executes successfully. It can be seen here
that one of the factors effecting the smooth transition from
component ‘i’ to component ‘j’ is the reliability of
component ‘i’ (denoted by RoCi). Hence exact probability
of transition can not be represented by MPoT. Instead
another matrix, called Imperfect “Probabilities of
Transition” matrix (denoted by MIonT) is computed using
values in MPoT and vector VRoC (VRoC is a vector that
holds the reliabilities of all the components) as shown in
equation 2 below.

\[ IPoT_{i\rightarrow j} = RoC_i \times PoT_{i\rightarrow j} \text{ for all i, j} \]  

Using the values in Imperfect State Transition matrix we
generate another matrix called Cumulative State
“Probabilities of Transition Matrix (MCiPoT). Value at a
specific location in MCIoT is sum of all the previous
values in that row.

In each simulation run, system starts its execution from
component 1. Then a uniformly distributed random
number is generated using a good random number
generator. Depending upon the number that has been
generated, it is decided to which component the transition
will take place from the current component.

Given that ‘i’ is the current component and a random
number RAND has been generated, if MCiPoT\(_{ij}\) < RAND
<= MCiPoT\(_{ij+1}\) we assume that transition to the jth
component has taken place and jth component becomes
the current component. This process continues until
TERM (Terminal state or component) is reached or the
process fails. If TERM is reached, operation is successful
otherwise failure. System reliability is then computed by
dividing the number of successful operations by total
simulation runs. So here System Reliability (which is the
probability of the system completing desired task) has
been mapped over the transition path ending in the
terminal state.

4. Algorithm Description

1. Estimate the parameters
   a). Reliabilities of all Components i.e. RoCi’s.
   b). Reliabilities of all Transitions i.e. RoC
   c). Interact (Ci, Cj), TNC \(i\rightarrow j\)
2. Compute probabilities of Transitions PoT \(i\rightarrow j\)’s
   using Eq. 1
3. a). Construct Reliabilities of Components”
   vector “VRoC” using RoCi’s.
b). Construct “Probabilities of Transitions” matrix $MPoT$ using $Poi_{i\rightarrow j}$’s.

4. Initialize counters $SCOUNT$ for successful termination of a particular Course-of-Execution & $UCOUNT$ for unsuccessful termination of a Course-of-Execution.

5. Read in $VRoC$, $MPoT$, $TRUNS$.

6. Compute Imperfect “Probabilities of Transitions” matrix $MIPoT$ using Eq. 2.

7. Compute Cumulative “Probabilities of Transitions” matrix $MCPoT$.

8. Repeat steps 9 to 11’ $TRUNS$ times

9. $i = 1$

10. Generate a uniformly distributed and independent random number $RAND$ (for randomly selecting a Course of Execution).

11. Select a Course-of-Execution as follows:
   If ($0 < RAND <= MCPoT_{i\rightarrow 1}$)
      $i = 1$;
      Continue to step 9.
   Else
      If ($MCPoT_{i\rightarrow 1} < RAND <= MCPoT_{i\rightarrow 2}$).
         $i = 2$;
         Continue to step 9.
      .
      .
      .
      .
      .
      .
   Else
      If ($MCPoT_{i\rightarrow (N-1)} < RAND <= MCPoT_{i\rightarrow N}$)
         $i = N$;
         Continue to step 9.
      Else
         If ($MCPoT_{i\rightarrow N} < RAND <= MCPoT_{TERM}$)
            $SCOUNT = SCOUNT + 1$ (Path terminates successfully).
         ENDIF
      ELSE
         $UCOUNT = UCOUNT + 1$ (Path terminates unsuccessfully).
      ENDIF

12. (Compute Application Reliability)

$$RELappl = \frac{SCOUNT}{TRUNS}$$

5. Simulator Implementation

Simulator developed for the purpose was applied in many ways. In the first case a sensitivity analysis was done, where effect of the reliability of each component on the overall reliability of the system was analyzed. In the second case we tried to find out effect of number of components on the reliability of a component based application.

5.1 Sensitivity Analysis

In this implementation a detailed sensitivity analysis was done. In any system, the overall output depends, to some extent, on every input. Here reliabilities of individual components are taken as inputs and overall system reliability is taken as output. Every component has some effect on the overall reliability of the application. Note from Figure 1 that transition can take place from any component to any other component depending upon the Course-of-Execution, but there is no transition from terminal component to any other component. Probabilities of transition for a four component system are shown in the table 1.

![Component Control Flow Graph](image)

Table 1: “Probabilities of Transition” values

<table>
<thead>
<tr>
<th>$Poi_{i\rightarrow j}$</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>TERM</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0.0</td>
<td>0.19</td>
<td>0.16</td>
<td>0.27</td>
<td>0.38</td>
</tr>
<tr>
<td>C2</td>
<td>0.21</td>
<td>0.0</td>
<td>0.24</td>
<td>0.40</td>
<td>0.15</td>
</tr>
<tr>
<td>C3</td>
<td>0.34</td>
<td>0.11</td>
<td>0.0</td>
<td>0.29</td>
<td>0.26</td>
</tr>
</tbody>
</table>
Given that VRoC = {.95, .93, .98, .97, 1.0 (for terminal component)}

System Reliability achieved in this case using above mentioned procedure with one Lakh (100000) simulation runs is .847.

Similarly the procedure was applied on different set of values. For finding effect of the reliability of each component on the overall reliability, reliability of that component was changed from .70 to 1.0 in steps of .5 each time. While doing so, the reliabilities of the other components were kept constant. This way the simulator was executed 100000 times for each combination. The results obtained are shown in table 2. The value in the table at each intersection of “component” and “reliability” depicts the Overall System Reliability for that value of the component reliability while other component reliabilities are kept constant. For example the value .562 at the intersection of second column and second row of the table is the overall system reliability when reliability of the component C1 is .70 while reliabilities of all other components are kept as 1.0.

Table 2: Overall System Reliability for various Reliabilities values for different Components

<table>
<thead>
<tr>
<th>Component</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>.562</td>
<td>.791</td>
<td>.439</td>
<td>.800</td>
</tr>
<tr>
<td>C2</td>
<td>.623</td>
<td>.821</td>
<td>.501</td>
<td>.826</td>
</tr>
<tr>
<td>C3</td>
<td>.688</td>
<td>.852</td>
<td>.578</td>
<td>.858</td>
</tr>
<tr>
<td>C4</td>
<td>.757</td>
<td>.885</td>
<td>.673</td>
<td>.890</td>
</tr>
<tr>
<td>.90</td>
<td>.831</td>
<td>.920</td>
<td>.796</td>
<td>.925</td>
</tr>
<tr>
<td>.95</td>
<td>.914</td>
<td>.960</td>
<td>.907</td>
<td>.962</td>
</tr>
<tr>
<td>1.0</td>
<td>.2097</td>
<td>.2098</td>
<td>.2090</td>
<td>.2096</td>
</tr>
</tbody>
</table>

As can be seen from the plot of table 2, i.e. Graph 1, overall reliability of the application increases when reliabilities of individual components are increased, keeping all other reliabilities constant. It can be of great help while choosing a component from some existing package or application for fitting it in a new application. A component can be accepted or rejected depending upon effect of its reliability on the overall reliability of the system.

5.2 Effect of no of Components on System Reliability

Second implementation is for checking the effect of number of components on the overall system reliability. System was simulated for 4, 5, 6, 7, 8 and 9 components respectively. In each category, number of simulation runs was kept One Lakh (100000). The results of this implementation have been summarized in table 3.
Table 3: Number of Components and System Reliabilities

<table>
<thead>
<tr>
<th>No of Components</th>
<th>Component Reliabilities</th>
<th>System Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>.95, .93, .98, .97</td>
<td>.847</td>
</tr>
<tr>
<td>5</td>
<td>.95, .96, .98, .97, .93</td>
<td>.806</td>
</tr>
<tr>
<td>6</td>
<td>.95, .96, .98, .97, .93, .95, .97</td>
<td>.778</td>
</tr>
<tr>
<td>7</td>
<td>.95, .96, .98, .97, .93, .95, .97</td>
<td>.758</td>
</tr>
<tr>
<td>8</td>
<td>.95, .96, .98, .93, .95, .97, .98</td>
<td>.751</td>
</tr>
<tr>
<td>9</td>
<td>.95, .96, .98, .97, .93, .95, .97, .98</td>
<td>.721</td>
</tr>
</tbody>
</table>

Graph 2: Number of Components v/s System Reliability

Data of table 3 is plotted in Graph 2 above, which shows the relationship between numbers of components in a component-based system and overall System Reliability. As can be seen, system reliability decreases as we increase the number of components in the system. It happens due to increase in the number of transition paths.

6. Discussion and Conclusion

In any large system, made up of subsystems, the overall reliability of the system depends a great deal on the reliability of each component of the system. Here, we tried to find out the impact of individual component reliabilities and transition probabilities on the reliability of the system seen in a larger perspective. Overall system reliability is sensitive to the reliabilities of individual components. In component-based applications, Components can be purchased off the shelf. If reliability of the component, being purchased off the shelf, is known in prior, its effect on the overall application reliability can be computed and a decision can be made whether to incorporate that component in the system or search for a different alternative. As regard to the number of components in a system, if they are increased, the overall system reliability starts decreasing. It may happen due to increase in the number of transition paths in the system. The simulation model proposed here can be of great help while designing and development, or composing software from existing software components. It may help in deciding how many components can be there in an application keeping in view how much reliable software is desired.

References:


Components,” In the proceedings of 8th International Symposium on Software Reliability Engineering,” 2nd–5th Nov. 1207, pp.146-155.


Dr. P.K. Suri received his Ph.D. degree from Faculty of Engineering, Kurukshetra University, Kurukshetra, India and master’s degree from Indian Institute of Technology, Roorkee (formerly known as Roorkee University), India. He is working as Professor in the Department of Computer Science and Applications, Kurukshetra University, Kurukshetra – 136119 (Haryana), India since Oct. 1203. He has earlier worked as Reader, Computer Sc. & Applications, at Bhopal University, Bhopal from 1985-90. He has supervised eleven Ph.D.’s in Computer Science and thirteen students are working under his supervision. He has around 125 publications in International/National Journals and Conferences. He is recipient of ‘THE GEORGE OOMAN MEMORIAL PRIZE’ for the year 1201-92 and a RESEARCH AWARD –“The Certificate of Merit – 2000” for the paper entitled ESMD – An Expert System for Medical Diagnosis from INSTITUTION OF ENGINEERS, INDIA. His teaching and research activities include Simulation and Modeling, Software Risk Management, Software Reliability, Software testing & Software Engineering processes, Temporal Databases, Ad hoc Networks, Grid Computing, and Biomechanics.

Sandeep Kumar received his Masters Degree in Computer Science from Department of Computer Science and Applications, Kurukshetra University, Kurukshetra, Haryana, India in 2001. He is a Ph.D. scholar under the guidance of Dr. P.K. Suri at Department of Computer Science and Applications, Kurukshetra University, Kurukshetra. He has more than seven years of teaching experience at institutions of repute. Presently he is working as Assistant Professor and Head, Department of Computer Applications, Dronacharya Institute of Management and Technology (DIMT), Kurukshetra since July 2007. Prior to this he worked as a lecturer at DIMT, Kurukshetra and Asia Pacific Institute of Information Technology SD India (APIIT SD India), Panipat, Haryana, India. He was the editor of Proceedings of an International Conference (CNFE’ 05) at APIIT SD India. His research interests include Component Based Software Engineering, Simulation, and Operating Systems.