Comparative Analysis of Software Reliability Estimation Models –State and Path Based

Arashdeep Kaur(Student)\(^1\) and Monika(Assistant Professor)\(^2\)
\(^1\)U.I.E.T, Panjab University, Chandigarh, U.T, India
\(^2\)U.I.E.T, Panjab University, Chandigarh, U.T, India

Abstract - Software reliability is an important factor that contribute to the quality of software. The objective of this paper is to provide an overview of the research in the area of architecture-based software reliability models considering the system architecture approach, uncertainty factors influencing the model. The aim of this paper is to establish a relationship between various models by identifying the limitations and enhancements of previous models over the new models.

Keywords: component-based software, reliability analysis, markov chains, component dependency graphs.

1 Introduction

Software reliability is often defined as “the probability of failure-free operation of a computer program for a specified time in a specified environment [23].” It refers to the quality of the software. Early quality prediction at the architecture design stage is highly desired by software managers and developers, as it provides a means for making design decisions and thereby facilitates effective development processes [1]. Software architecture analysis aims at investigating how an architecture meets its quality requirements [2] based on the structure and the correlation among the components of the software.

In Figure 1 evaluation of analytical models is categorized based on architecture and failure behavior. Hierarchical method and Composite method are the two approaches which can be used as a solution method in architecture based prediction for reliability. Prevalent approaches to software reliability modeling are black-box based [1], i.e., the software system is considered as a whole and only its interactions with the outside world are modeled, without looking into its internal structure. Several critiques of these time-domain approaches have appeared in the literature and some of these include the fact that they are applicable very late in the life-cycle of the software, ignore information about testing and reliabilities of the components of which the software is made, and do not take into consideration the architecture of the software.

White box reliability modeling is another approach that considers the internal structure of the software. The approaches to architecture-based prediction fall broadly into two categories: state-based approaches [7], and path-based approaches [13]. The path-based approaches to architecture-based software reliability prediction generally assume that the successive executions of the components are independent. This assumption leads to very pessimistic estimates of reliability, and largely impedes the applicability of these approaches.

2 State-based models

State-based models estimate software reliability analytically. They assume that the transfer of control between modules has a Markov property, that is, model software architecture with a discrete time Markov chain (DTMC), continuous time Markov chain (CTMC), or semi Markov process (SMP). The reliability of software application is estimated either by solving the composite model that combines software architecture with failure behavior (composite method), or by superimposing failure behavior on the solution of the architectural model (hierarchical method). If the reliability improves over time, as faults are discovered and corrected, one would expect that the number of failures detected per unit of time would be decreasing and the time between failures would be increasing.
Table 1. Summary of State based models with uncertainty factors.

<table>
<thead>
<tr>
<th>S. no</th>
<th>State Based Model</th>
<th>System Architecture</th>
<th>Uncertainty factors influencing the Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Littlewood (1975) [5]</td>
<td>Irreducible CTMC</td>
<td>Operational Profile, Time between Failure, Component Failure Rate</td>
</tr>
<tr>
<td>2.</td>
<td>Littlewood (1979) [6]</td>
<td>Irreducible SMP</td>
<td>Operational Profile, Time between Failure, Component Failure Rate</td>
</tr>
<tr>
<td>4.</td>
<td>Laprie (1984) [8]</td>
<td>Irreducible CTMC</td>
<td>Mean execution time of Component, Number of Failures, Time between failures</td>
</tr>
<tr>
<td>6.</td>
<td>Littlewood (1995)</td>
<td>DTMC</td>
<td>Total workload, Operational Profile, Time between Failure, Component Failure Rate</td>
</tr>
<tr>
<td>9.</td>
<td>Reussner (2003) [22]</td>
<td>DTMC</td>
<td>Individual Component Reliability, Number of Component Executions, Operational Profile</td>
</tr>
</tbody>
</table>

*DTMC(Discrete time markov chain)
**CTMC(Continous time markov chain)
***SMP(Semi-markov process)

Factors that bring uncertainty in reliability estimation based on Path-based models are the operational profile, Number of component execution, Individual Component reliabilities. Table 1 gives an overview of the various state based models together with the system architecture and uncertainty factors influencing the model.

3  Various Models

3.1  State Based Models

1. Littlewood Model [5] - [6] : This was one of the earliest approach to estimate software reliability. It considers software reliability in terms of operational reliability. Firstly a reliability system with system architecture based upon irreducible CTMC was made [5]. An another approach [6] was developed that consist of a modular program in which transfer of control between modules follow a SMP. This Model describes structure via dynamic behavior using Markov assumption. It analyze both the component and interface failures.

Limitations of Littlewood Model [6]:

- This model considers that failure occur if given input value specification of computation to be performed and output values are incorrect or delayed not considering performance requirements.
• Littlewood Verrall is applicable when there are no failures during testing or when failure data are not available.
• This model suffers from the shortcomings of the composite solution approach which considers both the architectural behavior and failure behavior together.

2. Cheung Model [7]: The early approach (for not continuously running applications) by Cheung uses a discrete time Markov chain (DTMC). The user-oriented reliability model developed to measure the reliability of service that a system provides to user. A simple Markov model is formulated to determine the reliability of a software system made up of n components by component failures is the special case of littlewood model [6]. Transitions between components follow a DTMC with initial state probability vector \( q = [q_i] \) and transition probability matrix \( P = [p_{ij}] \). The solution method taken in this work is hierarchical. The reliability of component is estimated as the probability that no failure occurs during its execution. This model was made for certain improvements in cheung model [7].

Enhancements in Cheung Model:
• It measures the reliability of software system with respect to user environment.
• various effects of user profile is discussed, which summarizes the characteristics of users of system on software reliability.
• Sensitivity analysis techniques are developed to determine the modules that are most critical to system reliability.

Limitations of this approach:
• The applicability of this model is restricted to two assumptions:
  • module reliability independence and transfer of control independence.
  • This model may be refined to be considered for component failures due to aging in addition to design errors.
  • The model shows that the components have to be extremely reliable in order to produce an acceptable results which is not always the case.

3. Laprie Model [8]: This model which considers only the component failures is the special case of littlewood model [6]. It describes the software system made up of n components by a CTMC.

Enhancements in Laprie Model:
• It considers component interconnection, inter component transition probability, component failures and other statistical information. It is assumed that each component fails with constant failure rate.
• Embedded DTMC of this \((n+1)\) state CTMC is equivalent to DTMC that represent Cheung Model [7] with additional transitions from both exit state and failure state to starting state which represent immediate reset/restart of program execution.

Limitations of this Approach:
• Laprie Model [8] assumes that the failure rates are much smaller than the execution rates which leads to asymptotic behavior relative to the execution process. To overcome this limitation Laprie model considers the hierarchical solution method.

4. Kubat: This model was considered as an enhancement over cheung model. The model proposed by Kubat [9] includes the information about execution time of each component, thus resulting in an SMP as a model of software architecture. Transitions between components follow a DTMC with initial state probability vector \( q = [q_i] \) and transition probability matrix \( P = [p_{ij}] \). The solution method taken in this work is hierarchical. The reliability of component is estimated as the probability that no failure occurs during its execution. This model was made for certain improvements in cheung model [7].

Enhancements in this Approach:
• Kubat [9] takes into account the execution time of each component for measuring the reliability. It provides an approach to measure the each component reliability as compared to cheung [7] in which no proper method was defined.

Limitations of Kubat model:
• It concludes that component failure leads to system failure. some measures are still needed to be taken to avoid this problem.
• Kubat is based on the assumption that components are highly reliable and variances of number of times each component is executed are very small. This cannot be assumed for all types of softwares.

Note that once component reliabilities are estimated the solution approach reduces to the hierarchical treatment of the Cheung model [7].

5. Gokhale [10] describes the architecture by an absorbing DTMC and uses a hierarchical solution method. However, it differs in the approach taken to estimate the component reliabilities. Given time-dependent failure intensity \( \lambda_i(t) \) and the cumulative expected time \( V_i \) spent in the component per execution of the application, the reliability of component is estimated as

\[
R_i = e^{- \int_{0}^{V_i} \lambda_i(t) \, dt}
\]

Enhancements in Gokhale model:
• It provides an enhanced approach for estimating component reliability considering time dependent failure rates and utilization of component through cumulative expected time spent in component per execution.
• It donot assume that component reliability of software architecture is given. It can be obtained experimentally by testing the application.
Limitations of Gokhale model:

- Gokhale concludes that Component failure leads to system failure. This is not a valid statement depending upon whether the part containing bug is executed or not. There must be some approach related to this issue that is needed to be considered.
- Solution to issue of dependency among components was not developed. This is the major drawback in all the previous models as well as Gokhale model.

Note that, the special case of this model that assumes constant failure intensities is equivalent to the special case of Kubat model [9] that assumes deterministic execution times.

6. Ledoux [11]: A general model (Op) is presented and is specifically designed for software systems; it allows the evaluation of various dependability metrics, in particular, of availability measures. Op is an attempt to overcome some limitations of the well-known Littlewood [6] reliability model for modular software.

Enhancements in Ledoux model:

- Ledoux considers the way failure processes affect the execution and deals with the delays in recovering an operational state.
- Primary failure and Secondary failures was discussed, which was one of the limitations in Littlewood model.

Limitations of Ledoux model:

Analysis of factors affecting the reliability was not considered.

7. Reussner Model [22] uses Markov chains to model system architecture. This method uses the idea to express component reliability not as an absolute value but as function of the input profile of the component. Markov chains are constructed in a hierarchical manner and states include calls to component services in addition to usual component executions. Services may invoke different methods, which may be either internal or external for the component. The reliability of the component is calculated by the reliabilities of the methods, which it uses.

3.2 Path-based models

Path-based models are similar to state-based models, but consider only finite number of component executions traces. It usually correspond to system test cases.

Table 2: Summary of Path based models with uncertainty factors

<table>
<thead>
<tr>
<th>S. no</th>
<th>Path based Model</th>
<th>System Architecture</th>
<th>Uncertainty Factors Influencing the Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shooman (1976)</td>
<td>Markov Structure</td>
<td>Number of bugs in software, system operation time.</td>
</tr>
<tr>
<td>5</td>
<td>Hamlet (2001) [16]</td>
<td>Component call Graph</td>
<td>Probability of occurrence of each subdomain, Operational profile, Individual component reliability.</td>
</tr>
</tbody>
</table>

1. Shooman model [12]: This is one of the earliest models that considers the reliability of modular programs, introducing the path-based approach by using the frequencies with which different paths are run.

Based on the assumption that software execution can take fixed number of different Paths. The frequency of occurrence of each path and its failure probability are assumed to be known.

Limitations of Shooman model:
• Dependency among components is not considered as it is important issue to be discussed.
• It donot provide the solution of execution of different paths.
• Some parameters are assumed to be known. No method is defined for there evaluation.
• Shooman assumed that failure rate(crash rate) is directly proportional to the number of remaining errors. But this is not valid because two bugs in less frequently occurring code is more reliable than the one bug in frequently occurring one.

2. Krishnamurthy and Mathur model [13] takes an experimental approach to obtain path reliability estimates, a sequence of components along different paths are observed using the component traces collected during testing called Path Traces. Assuming that individual components fail independently of each other, it follows that the path reliability is the product of components reliabilities.

Enhancements in Krishnamurthy model:
• Krishnamurthy addresses the problem of intra component dependency(in case of loops).
• Seeding fault method is applied to obtain component reliability(ie component and interfaces are identified).
• Information collected using path traces was assumed to be given in shooman [12].

Limitations of Krishnamurthy model:
• This approach doesnot consider component interface-errors although they are considerable factors in reliability analysis of component based software.
• Estimating reliability based on test cases don’t take into consideration the frequency of interaction between components.

3. Yacoub, Cukic and ammar Model [14] [21] takes an algorithmic approach to estimate path reliabilities; a tree-traversal algorithm expands all branches of the graph that represents software architecture. The breadth expansions of the tree are translated as the summation of reliabilities weighted by the transition probability along each path. The depth of each path represents the sequential execution of components, and is hence translated to multiplication of reliabilities.

Enhancements in this approach [14]:
• Analysis based on execution scenarios taking into account algorithmic approach.
• Probabilistic model name CDG is constructed.It guarantees that the loops between two or more components donot lead to a deadlock.
• Using CDG model, we can incorporate effect of frequently executed components,interfaces and links.
  Hence we can dedicate more testing and development effort to those critical artifacts.

Limitations of Yacoub (SBRE) model [14]:
• The nature of the application: The approach is applicable to component-based application which are analyzed using execution scenarios.
• The algorithm can be used for sensitivity analysis of the application reliability to the variation in the component and interface reliabilities in a given period of execution.
• It does not consider failure dependencies between components.

An Yacoub (SBRA) model [21] have some enhancements:
• Develop a reliability analysis technique that addresses issues related to the distributed nature of software systems, such as the complexity and hierarchical composition of subsystems.
• Algorithm application results in identifying critical components, subsystems, and links which require increased attention in testing, verification, and validation.

Limitations of this approach:
The algorithm does not consider the overall application reliability growth as a function of time. Further, some scenarios may be more critical than others, but they are seldom executed.

4. Hamlet model [16] is also regarded as pathbased, as it considers the actual execution traces of component execution given the mapping from input to output profile. This model tries to address the issue of unavailability of component's usage profile in early system development phases. To do so do not assume fixed numeric values for reliability but provide model mappings from particular input profile to reliability parameters.

Different input profiles are represented by dividing the input domain of the component to sub-domains and assigning a probability for each sub-domain to occur. This model does not consider explicitly the architecture of the system. Instead, it calculates the output profile of a component, which actually is the input for the next component and is used to calculate latter reliability.

4 Discussion
Although the path based approaches represent the failure behavior of the components using the probability of failure or reliability, the state-based approaches allow component failure behavior to be represented using three types of failure models, namely, probability of failure or reliability [7], [18], [3], [4 constant failure rate [19], and time-dependent failure intensity [20]. The difference in reliability predictions of the statebased and path-based approaches becomes evident only when the control flow graph of the application contains loops. Thus, while state-based models analytically account for the potentially infinite number of paths, pathbased models restrict the number of paths to ones observed experimentally during the testing or terminate the depth traversal of each path using the average execution time of the application [14].
In Path based models, the method of combining software architecture with components and interfaces failure behavior is not analytical. First, the sequence of components executed along each path is obtained either experimentally by testing [13] or algorithmically [14] and the path reliability is obtained by multiplying the reliabilities of the components along that path. Then, the system reliability is estimated by averaging path reliabilities over all paths.

5. Conclusion

The framework presented in this paper addresses the enhancement and limitations of the architecture based reliability models. We have classified the these models based on the system architecture approach and the uncertainty factors influencing these models. A notable drawback of path-based approaches is that they provide only an approximate estimate of application reliability when the application architecture has infinite paths due to the presence of loops. Based on the solution methods of architecture based prediction the reliability of software is estimated but still many enhancements are required in the models to overcome the limitations that occurred in the above approaches.

6 References


