

A Comparative Study On Reliability And Cost Estimation For Architecture-Based Software

Srinivasarao.Sabbineni ¹ and Dr. Kurra.Rajasekhara Rao ²

*Assistant Professor, Computer Science and Engineering Department,
K.L.University, Vaddeswaram, Guntur, AP-522502, India
Srinu1479cse@kluniversity.in
Director SRI PRAKASH COLLEGE OF ENGINEERING
Rajupeta, Tuni-533401, E.G.Dist, AP.
krr@sriprakash.org*

ABSTRACT:

The purpose of this paper is to provide an overview of the existing research in this area. Software reliability is defined as the probability of the failure free operation of a software system for a specified period of time in a quantified environment. 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 allocated reliability and cost.

Keywords: Architecture-based Software, Software Reliability, Reliability allocation, Dynamic programming, Software development, software model, Reliability Estimation.

1. INTRODUCTION:

Information technology is playing a important role in existing society.it has changed the whole world into a universal village with a global economy. Today almost every business, industry, services, government agencies, and even our day-to-day activities are directly or indirectly affected by computing systems. The computer revolution has profited society and increased the global efficiency, but a major threat of this revolution is that the world has become critically dependent on the computing systems for the proper working and effectiveness of all its activities.

Some of the important Examples are air traffic control, nuclear reactors, patient monitoring system in hospitals, in motorized mechanical and safety control, online railways and air ticketing, industrial process control, global networking of various business, and services which include information storing (databases),

information sharing and internet marketing, etc. If the computer shows a failure in such systems in impact of failures may range from inconvenience in social life to economic damage to defeat of life in the most critical case.

The rest of this paper is organized as follows. In Section 2, we introduce in detail the architecture-based model and In Section 3, we introduce in detail the software architecture and its styles. In Section 4, software reliability and cost model and a numerical example is given in order to validate whether the reliability and cost model can work properly. In Section 5 conclusion and future work are presented.

2. ARCHITECTURE-BASED MODELS:

Architecture-based approach can be used to calculate system reliability from components information, or to calculate system reliability of components. These models put emphasis on the architecture of the software and derive reliability estimates by combining estimates obtained for the different models of the software.

The architecture-based software reliability models are classified into three types. They are:

1. state-based models
2. path-based models
3. Additive models

2.1.: State-based models:

This discussion of models uses the control flow graph to represent the architecture of the system. It is expected that the transfer of control between modules has a Markov property which means that given the knowledge of the component in control at any given time, the future behaviour of the system is conditionally independent of the past behaviour. The architecture of software has been demonstrated as a discrete time Markov chain (DTMC), continuous time Markov chain (CTMC), or semi-Markov process (SMP). These can be further classified into fascinating and difficult.

2.2. path-based models:

This class of models is constructed on the same common steps as the state-based models; except that the method taken to combine the software architecture with the failure behaviour can be described as a path-based since the system reliability is calculated bearing in mind the possible execution paths of the program either experimentally by challenging or algorithmically.

2.3. Additive models:

This class of models does not reflect explicitly the architecture of the software. Relatively, they are focused on estimating the whole application reliability using the component's failure data. It should be well-known that these models consider software reliability growth. They are called additive models since under the assumption that component's reliability can be modelled by NHPP the system failure intensity can be expressed as the sum of component failure intensities.

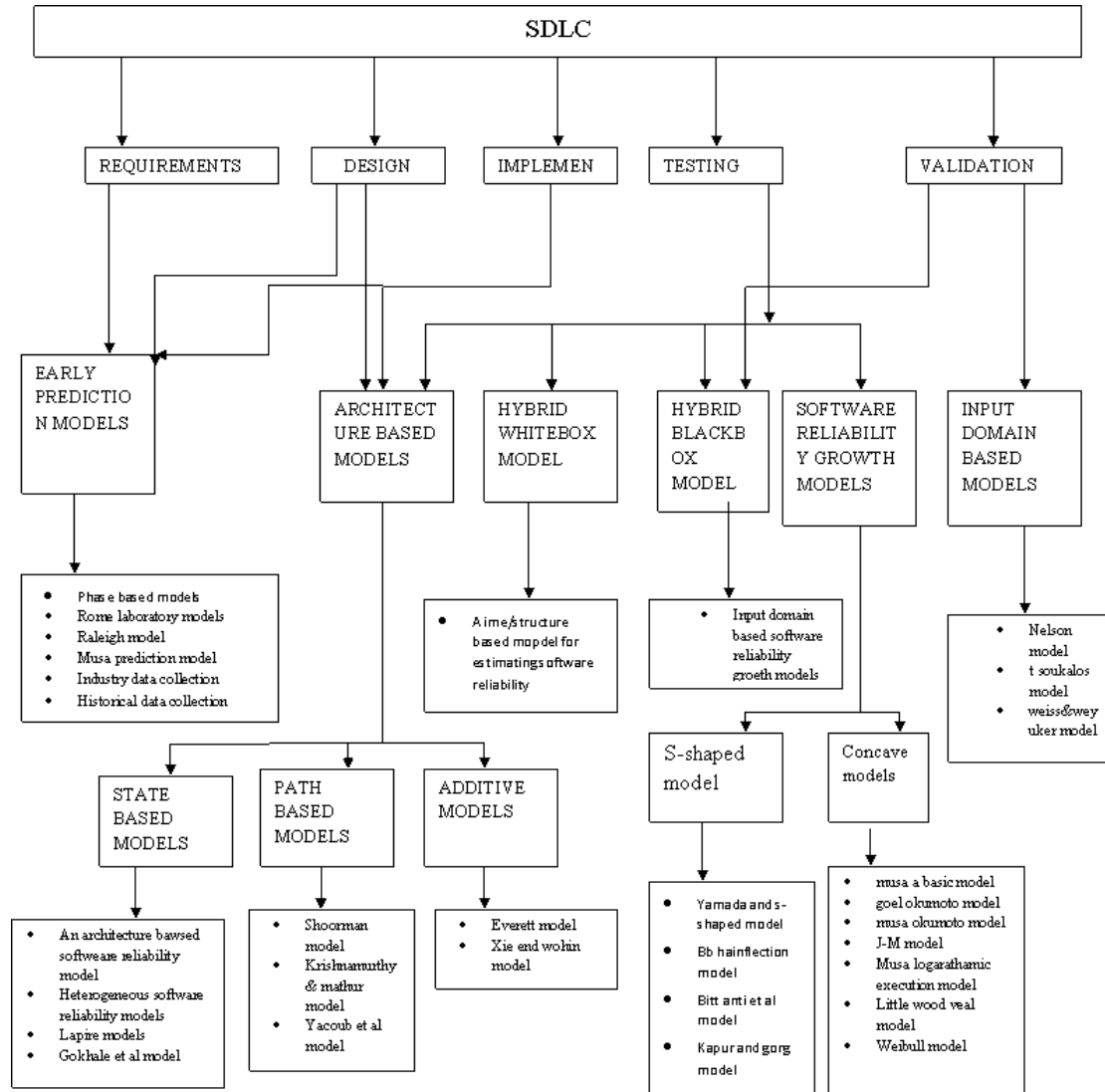


FIG 2.1: Software reliability model classification and selection model

3. SOFTWARE ARCHITECTURE:

Software architecture, which describes the structure of software at an abstract level consists of a set of *components*, *connectors*, and *configurations*. Furthermore, a pattern that characterizes the configurations of components and connectors of software architecture is considered as architectural style.

Four architecture styles are used to demonstrate the development of the state machine. These styles include:

1. Batch-sequential
2. Parallel/Pipe-filter
3. Fault Tolerance
4. Call-and-Return

3.1.: Batch-Sequential:

In a batch-sequential style, components are implemented in a sequential manner, i.e., only a single component is executed in any occasion of time. For example, a bank executes a batch of transaction updates to a master file in sequence. From this type of architecture, the control is transferred to one (and only one) of its beneficiaries upon the completion of a component. The selection of the successive component is either probabilistic (if existing more than one successor) or deterministic (if having exactly one successor).

3.2.: Parallel/Pipe-Filter:

Parallel and pipe-filter styles are frequently used in concurrent implementation environments, in which a set of components is commonly executed concurrently to improve performance. These two styles are quite parallel; the main difference is that parallel computation is normally in a supportive multi-processors environment, whereas the pipe-filter style is mainly in a single processor, multiprocessors environment.

3.3.: Fault Tolerance:

A fault-tolerant architectural style is frequently adopted to increase availability of software. Typically, this style contains of a set of fault forgiving components, including primary components and hold-up components, in which the backup components reimburse for the failures of the crucial components. When a crucial component fails, one of its backup components promptly takes over the responsibility and becomes a new crucial component. If the new crucial component also fails, another backup component will momentarily take over. This scenario continues as long as there is a backup component available to keep software up and successively.

3.4.: Call-And-Return:

Call-and-return style is one of the most ubiquitous designs for both individual and distributed systems. The developments began from procedure calls, client-server architecture, to object-oriented message passing mechanism. In this style, a calling component (caller) can invoke a called component (callee) a number of times, and each time the callee proceeds the requested services back to the caller. For both invocation, the caller executes a context switch by allowing a temporary control to its callee and then waits for its reply. Once receiving the reply, the caller resumes the execution from where it missing. Ultimately, the caller transfers its complete control authority to its succeeding component.

4. SOFTWARE RELIABILITY AND COST MODEL:

According to the definition of software reliability

$$r(t) = e^{-\lambda t} \text{ -----(1)}$$

Where $r(t)$ is continuous-time system reliability. And λ is its failure rate. Every software is depends on its cost and its reliability. Reliability increases then cost of software development cost also increases.

See the fallowing assumptions A and B.

Assumption A):

The number of faults existed in software system is inversely proportional to system development cost, and system failure rate is also proportional to the number of faults.

The relationship between them can be written as

$$E \propto 1/C \quad \text{----- (2)}$$

$$\lambda \propto E \quad \text{----- (3)}$$

Where E is the number of failures and C is the development cost. The above eq-(2), reliability function can be written as

$$C \propto -1 / \ln r \quad \text{----- (4)}$$

Assumption B):

Software system development cost is always proportional to the complexity and size of the software system.

By taking assumption (B) into account, the above equation can be expressed as:

$$C = -\alpha / \ln r \quad \text{----- (5)}$$

Where in the above equation ‘ α ’ represents the complexity and size of the software and reliability coefficient in this paper. And assume in this paper ‘ β ’ represents the basic development cost. Development cost means personal training, development tools preparation etc.

With the assumption above, we describe the reliability-cost model as below:

$$C(r) = -\alpha / \ln r + \beta \quad \text{----- (6)}$$

The above equation (6) is relation ship between the system development cost and the total system reliability in our model.

4.1 Numerical example of the RCR model

By assume $\alpha = 0.4$ then, and basic development cost $\beta = 90$ then reliability cost table is Consider a hypothetical example where we wish to estimate the overall development cost C , taking into account the two parameters, which are $\beta = 90$ and $\alpha = 0.4$; then we can obtain the data shown in Table 4.1 according to the formula of the RCR model. Fig. 4.1 shows the relationship between the system reliability and the development cost of the RCR model with the data in Table 4.1, where r and C refer to the software reliability and the development cost, respectively.

Table 4.1 Reliability and cost data set with $\beta = 90$ and $\alpha = 0.4$

S.NO	R	C
1	.90	93.79
2	0.91	94.24
3	0.92	94.79
4	0.93	95.51
5	0.94	96.46
6	0.95	97.79
7	0.96	99.79
8	0.97	103..13
9	0.98	109.79
10	0.99	129.79

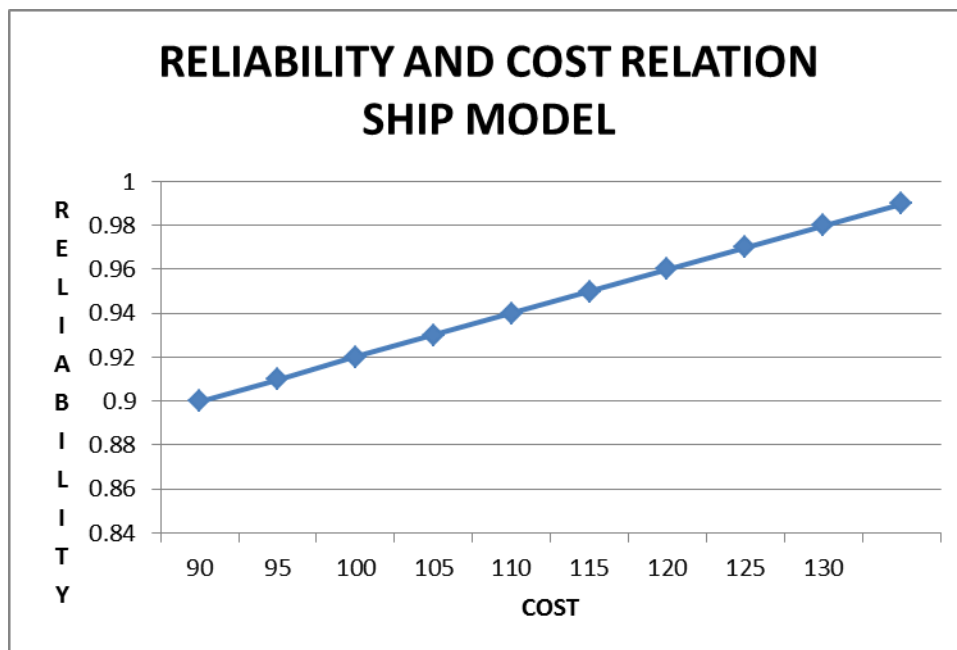


Fig.4.1 C and r for the hypothetical example

5. CONCLUSION AND FUTURE WORK:

In this paper, reliability and cost estimation approach model for architecture-based models and different architecture styles are covered. At end we had taken the relationship between software reliability and software development cost taking into account. The outcome of this work, despite being still preliminary, allows us to say that the method proposed here is a comparatively straight advancing one in conveying the relationship between reliability and cost during design phase. Future work is one is how to allocate the limited cost and reliability for particular component and second is identifying the failure of particular component in early stages of software development and asses the risk levels in architecture-based software so that losses may reduce in early stages.

6. ACKNOWLEDGEMENTS

I state that in my research work, I am being guided by Dr.Kurra.Rajasekhara Rao Director, Sri Prakash College Of Engineering, Rajupeta, Tuni, E.G.Dist, Ap, and his guidance and support, I submitting research paper, which is part of research. I also acknowledge the support and encouragement given by my family members and my friends and well-wishers to pursue the Research work.

7. REFERENCES:

- [1] Hui Guan, Wei-Ru Chen, Ning Huang, Hong-Ji Yang “Estimation of Reliability and Cost Relationship for Architecture-based Software” *International Journal of Automation and Computing* 7 (4), November 2010, PP. 603-610.
- [2] Hui Guan, Tingmei Wang, Weiru Chen “Exploring Architecture-Based Software Reliability Allocation Using a Dynamic Programming Algorithm” *Inproc.2nd symposium Int’l computer science and computational technology, Huangshan, P.R.China, 26-28, Dec.2009, pp.106-109.*
- [3] D.L.Parnas. “The Influence of Software Structure on Reliability”. In *Proc.1975 Int’l Conf. Reliability software, Los Angeles, CA, April 1975. pp. 358-362.*
- [4] M.L.Shooman. “Structural models for software reliability prediction”. In *Proc. 2nd Int’l Conf. Software Engineering, San Fransisco, CA, October 1976, pp. 268-280.*
- [5] M.E. HELANDER, M. Zhao and N. Ohlsson. “Planning Models for Software Reliability and Cost”. *IEEE Trans. on Software Engineering*, 1998, 24 (6):420~434.
- [6] F. Zahedi and N. Ashrafi, “Software Reliability Allocation Based on Structure, Utility, Price and Cost”. *IEEE Trans on Software Engineering*, 1991, 17 (4):345-356.
- [7] B. Boehm, R. Valerdi, J A. Lane et al, *COCOMO Suite Methodology and Evolution, CrossTalk*, 2005, pp. 20-25.

- [8] C. Y. Huang, J. H. Lo and S Y. Kuo, "Optimal Allocation of Testing resource Considering Cost, Reliability, and Testing Effort", In Prof. 2004 Pacific Rim Dependable Computing, French Polynesia, 2004, pp.103-112.
- [9] S. Y. Kuo, C. Y. Huang and M R. Lyu, "A Framework for Modeling Software Reliability, Using Various Testing Efforts and Fault Detection Rates". IEEE Transactions on Reliability, 2001, 50 (3):310-320.
- [10] Srinivasarao.Sabbineni, Kurra.Rajasekhara Rao "Assessment of Architecture-based Software System Reliability Allocation on components using a Dynamic programming". IJCA, 2013, 73 (19):41-45.
- [11] SrinivasaRao.Sabbineni, RajasekharaRao.Kurra "Estimation of Reliability On Components using a Dynamic Programming". IJCSI, 2013, 10 (3):78-81.
- [12] Srinivasarao.Sabbineni, Dr.Rajasekhaarao.K, Kiran.V.D"Analysis of software reliability and cost assessment relationship for architecture-based software" GJCAT, 2011, 1 (3):403-406
- [13] A. Mettas, Reliability allocation and optimization for complex systems. In Proc. Annual Reliability and Maintainability Symposium, Los Angeles, CA, January 2000, pp.216-221.
- [14] R. W. Bulter and G.B. Finelli, "The infeasibility of quantifying the reliability of life-critical real-time software", IEEE Trans. on Software Engineering, 1993, 19:3-12.
- [15] M.R.Lyu. Handbook of Software Reliability Engineering. IEEE Computer Society Press, New York, 1996, pp.36.
- [16] M. R. Lyu. Handbook of Software Reliability Engineering. IEEE Computer Society Press, New York, 1996, pp.315