The Research Monograph Series in Computing, Electrical & Communication Networks
Vol. 1 (01), April 2023, pp. 30–41

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 INTRODUCTION

This thesis centers on the automated generation of performance feedback in software architectures with the purpose of interpreting the results of the performance for the analysis. The research progresses on by suggesting most suitable architectural reconfigurations methodology that has been devised by keeping track of the performance knowledge.

5.2 THROUGHPUT UTILIZATION ANALYSIS

In Figure 5.1, the throughput is reported and utilization obtained is shown in Figure 5.2, These refer to CPU and DISKS with a fixed thinking time of z = 1s while the number of clients N in the system. It can be noted that the same numerical results are obtained which supports the correctness of the transformation of the description into the queuing network model. The two tower solver (Marco Bernardo et al., 2011) which is unable to solve models with more than seven clients has results tackled as two towers. This is because the state space explosion phenomenon encompass during the time when the solvers handles the continuous time-Markov chain model.

5.3 PERFORMANCE ANALYSIS

In this section, the performance modeling approaches are described and developed in order to assess real time performance, task execution time as a function of the number of system interfaces and the thruster allocation convergence characteristics. The impact of the number of the sensors and thrusters on the control loop execution time distribution is studied by the stochastic analysis model. The thruster allocation model is executed through the actual implementation approach inside a performance model to assess the impacts of the number of thrusters on the convergence characteristics of the thruster's allocation library. It is because of the thruster allocation algorithm which is the most critical

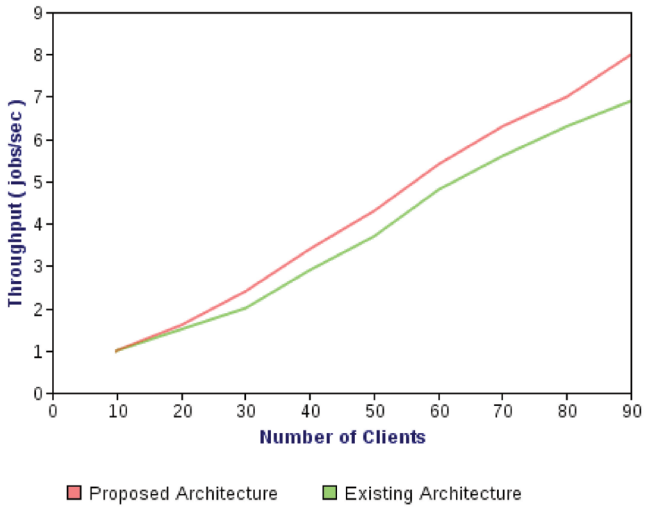

Figure 5.1 Graph for analyzing throughput.

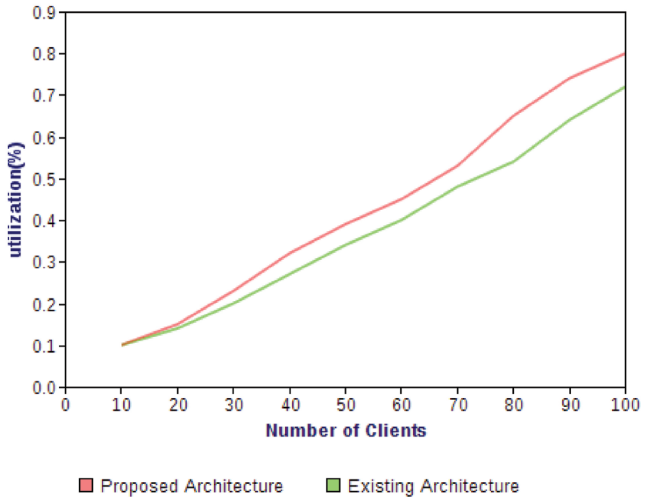

Figure 5.2 Graph for utilization indices.

task in the control loop. Experience with model based Performance, Reliability, and Adaptability Assessment of a complex industrial architecture will help out in solving the critical tasks.

5.3.1 Analysis of Real-time

Reasoning about time and temporal characteristics of the system are allowed by the real time. A 1s period is chosen as a cycle execution.

Experiments chosen as a cycle execution, Experiments which are conducted with a prototype execute each of the tasks independently and individually. It measures the execution time of a task for each instance executed.

5.3.2 Analysis of Stochastic Performance

Execution time distribution and its study provides additional insight into the timing behaviors of the system and it allows to estimate the quickness of the control loop execution in the most cases (eg: 95% of all cases). Rare misses of the systems, less conservative estimates for performance through stochastic analysis is to avoid over sizing of resources.

In order to show that the control loop deadline is satisfied in the worst case, the current system design the results from the earlier analysis is shown. Additional results related to the sensitivity of the control loop execution time to the vessel configuration is provided by the analysis in this subsection.

5.3.3 Analysis of Sensitivity

By using the parameterized architecture performance models the impact of components on the tasks execution times is analyzed in this section. There are two extension scenarios which are shown here.

Increase of output data and message which is the results of new types of diagrams or design added for further calculations. These are varying number of thruster of the vessel. A sensitivity analysis is performed both extension scenarios by first revalidating the real time performance and determining time executions by running the system simulation. More data retrieval tasks need to be executed on the performance level of when new types of design are added to the system.

5.4 ANALYSIS WITH RELIABILITY MODELING

Beginning from the UML diagrams it shows in this section, to build a reliability model as it is considered a target reliability on-demand of a component based system as function of: (1) The reliability of the software components and connectors. (2) The operational profile – the portability of invocation of use cases and the inclusion of the number of invocations of components and connectors (UML) profiles for non-functional properties at work: analyzing reliability availability and performance.

The probability of failure on demand in the original model is expressed as follows:

$$\varphi_s 1-\sum_{m=1}^{M} P_k \left(\prod_{i=1}^{Q} (1-\varphi_i)^{bpik} \prod (1-\Psi_{ij})^{\text{interact}(i,j,k)} \right) \tag{5.1}$$

where K is the number of system scenarios;
P_k is the probability of execution of scenario k;
φ_s is the failure probability on demand of the whole system;
φ_i is the failure probability on demand of a software component i;
bpik is the number of busy periods (i.e. invocations) of component i within scenario k;
N is the number of software components;
ψ_{ij} is the failure probability on demand of a software connector between components i and j;
interact (i; j; k) is the number of interactions between components i and j within scenario k (i.e. the number of times the connector between these two components is used);

Obtaining such model from annotated UML diagrams has been illustrated here. Towards making the transformation more explicit and to produce a model based on reliability notation Equation (5.1) has been reformulated here as Fault Tree.

Nodes of a Fault Tree are the events and logical operators having the root contains an undesired effect. The event which could cause this effect is added to the tree as a series of logic expressions.

5.5 MATHEMATICAL COMPLEXITY AND NONLINEARITY OF SYSTEM

Architecture based reliability evaluation of software intensive systems are the outcome of the development of a number of the development of a number of models and mathematical functionalities. Broadly speaking architecture based reliability evaluation is treated as mathematical functions obtained from architecture annotations and configurations to reliability metrics like mean time to failure (MTTF) failure rate or failure portability. Estimated parameters are included in the input the

Table 5.1 Mathematical formulae.

Quality Attribute	Quality Model
Reliability	Reliability Block diagrams [24,98,109,126,134,141]
	Markov Chains [15,34,39,144]
	Fault Trees [32,110,125,140]
	Dependency Graphs [101,112,119,136]
Performance	Aggregation Functions [74,91,51]
	Markov Chains [88,92]
	Queuing Networks [115,126,150]
Energy consumption	Aggregation Functions [71,74,56]
	Markov Chains [37,46,51]
	State machines [67,94,118,139]

function. This in turn includes execution of initialization probabilities, transition probabilities among software components, hardware failure rates, software failure rates and software failure rates. Certain important information on structured and behavioral aspects of the system (eg: dependences of failures among components) are ignored by the use of simple aggregation functions or linear mathematical formulae in Table 5.1 and is also presented in Section 2.2. Many researchers including Gokhale et al., (2009) have optimal and have pointed out for the requirements of sophisticated mathematical formulations for an accurate prediction of reliability of a software intensive system, keeping in view its operational and failure behavior. The comprehensive reliability have been successfully adopted and used by Markove chains. The reliability metric becomes a composite effect of many parameters rather than being linear function, decomposed into individual relationships with the composition of structural and behavioral aspects into Markovian reliability modules. A series of complex mathematical operations such as application matrix and vector operators are required by the reliability evaluation models. Therefore the mathematical function obtained from the input parameters to the reliability metric is neither a linear nor simple aggregation function. So, an architect may find it hard to lack propagate and effect on the reliability metric to individual parameters.

5.6 ANALYSIS OF TRANSFORMATION

Table 5.2 summarizes the parameterization of the Queuing Network Model for the ATM case study. The input parameters of the QN are

Table 5.2 Input Parameters for queuing network model in ATM system.

Service centre	Input parameters	
	ATM	
	Class A	**Class B**
LAN	44 msec	44 msec
WAN	208 msec	208 msec
webServerNode	2 msec	4 msec
libraryNode	7 msec	16 msec
controlNode	3 msec	3 msec
db cpu	15 msec	30 msec
db disk	30 msec	60 msec

Table 5.3 Response time requirements for ATM software Architectural Model.

Requirement	Required value	Predicted Value
Print balance	1.2 sec	1.5 sec
Pin change	2 sec	2.77 sec

reported: the first column contains the service center names, the second column shows their corresponding service rates for each class of job, i.e. **ClassA** and **ClassB**.

Table 5.3 shows the performance analysis results of the ATM queuing network model: the first column contains the name of requirements, the second column reports their required values, their predicted values are shown in the third column as obtained from the QN solution. It can be observed that a response time is owned by both services and it does not fulfill the required ones: The print balance service has been predicted as 1.5 sec, whereas the pin change service has been predicted as 2.77sec.

Table 5.4 shows the performance analysis results are obtained through the solution of QN models of the new ATM systems (that is **ATM1**, **ATM2**, **ATM3**) are tried and compared them with the obtained results from the analysis of the initial systems (i.e., **ATM0**). The response time of the pin change service is 2.18 sec, 1.6 sec and 2.24 sec across the different reconfigurations of ATM architectural model.

- **Spearman Rank Correlation**
  Spearman's rank correlation of coefficient is used for identifying the total amount of strength of correlation among the data set of

Table 5.4 Performance analysis results.

Requirement	Required Value	Predicted Value			
		ATM 0	ATM 1	ATM 2	ATM 3
Print balance	1.2 sec	1.5 sec	1.14 sec	1.15 sec	1.5 sec
Pin change	2 sec	2.77 sec	2.18 sec	1.6 sec	2.24 sec

Table 5.5 Spearman correlation of rank.

	Attribute	Applicability
Spearman's rho Pearson correlation	1.000	0.664
Sig(2-tailed)	–	0.014
N	11	11
Applicability Pearson correlation	0.664	1.000
Sig(2-tailed)	0.014	–
N	11	11

two variables, and check whether the value of correlation is either positive or negative (Blalock 1960).

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2 - 1)} \tag{5.2}$$

5.7 REDESIGNING PROCESS RESULTS

A collection of performance indices refers the output of the performance evaluation stage. The users provide the actual required values. The comparison is carried out when the results are identified from the evaluation of the performance model with the requirements provided by the customers. When the calculated requirements fail to meet the customer's requirements, the design changes and the feedback process starts the relationship between the two values is found by calculating spearman's coefficient and the attributes of interdependency of performance are identified.

By comparing the predicted value with the actual value the actual requirement changes to improve design is indentified by a mathematical equation. It is used due to the easy calculation and understandability. Thus the rules are generated on the values derived from the calculations.

$$P_{ijk} = \beta + \gamma_j + \psi_i + p_{ij} + \rho_{ijk} \tag{5.3}$$

where, P_{ijk} is a matrix of performance indices observations (with row index *i*, column index *j*, and repetition index *k*).
β is a constant matrix of the overall mean of performance indices.
γ_j is a matrix whose columns are the deviations of each performance indices (from the mean value β) that are attributable to the architecture model. All values in a given column of $\boldsymbol{\gamma}_j$ are identical, and the values in each row of γ_j sum to 0.
ψ_i is a matrix whose rows are the deviations of each performance indices (from the mean value β) that are attributable to architectural model. All values in a given row of ψ_i are identical, and the values in each column of ψ_i sum to 0.
p_{ij} is a matrix of interactions. The values in each row of p_{ij} sum to 0, and the values in each column of p_{ij} sum to 0.
ρ_{ijk} is a matrix of random disturbances.

Deriving a simple methodology is to be used for the performance analysis process and generate simple rules to provide feedback at the designed level. After analyzing the relationships between various performance indices the rules are generated. Towards the evaluation, the research work considers some example architectures design. In the Table 5.6 the improvements achieved by the generated rules in the example is summarized.

Calculated response time is shown in the Figure 5.3 in each redesign round which describes that the response time gets reduced frequently after the application of the rule.

The response time for each iteration for refactoring and applying the proposed rules, has shown a better improvement in response time. The

Table 5.6 Response time improvements through application of Rules.

Application	No. of Rounds	Initial Response Time (ms)	Improved Response Time (ms)	Percentage of Improvement in Response Time
ATM machine	4	5678	1997	65.63
CT scan	7	7185	2498	62.10
Railway Reservation System	13	5452	1931	68.25
Online Banking Application	12	9564	3465	61.35

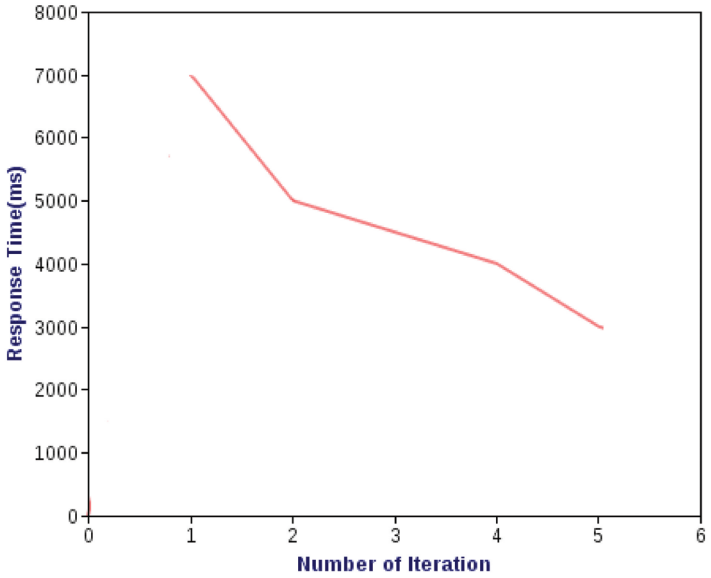

Figure 5.3 Improvements of response time by applying rules.

researcher has refactored the design and has shown sign of improvement in the design which is shown in Figure 5.3. The performance after each iteration is shown in Figure 5.4.

5.8 VARIOUS PERFORMANCE PREDICTION METHODS

There have been quite a few numbers of researchers towards developing a method of an application and including to improve the quality of the software design.

Table 5.7 summarizes the different researchers moving towards improving the functional requirements of a system by using the method of rule generation and application.

It has been revealed in an analysis of the literature that the evidence of the existence of a framework combining all the steps of performance analysis has improvement. It also state that there is a very few researchers towards the direction of application of performance tuning to component based systems. This research work stands as unique through the introduction of methodology for design change identification and for suggestions to design changes towards improving performance, maintainability and reusability. The comparison of various methods is shown in the Figure 5.5.

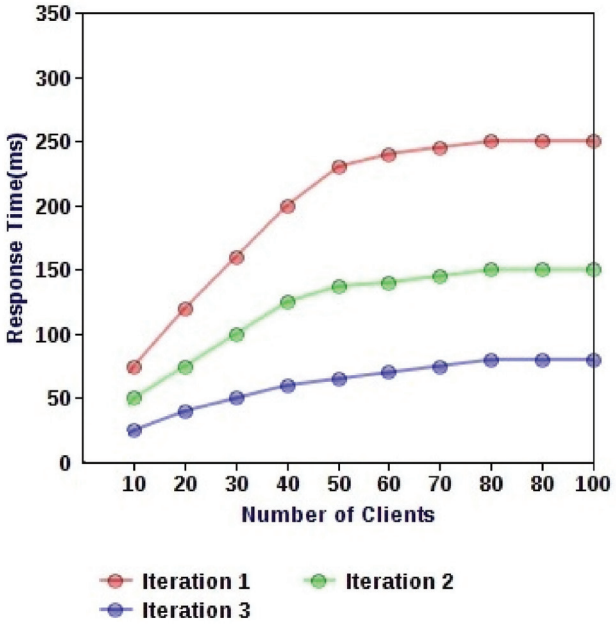

Figure 5.4 Comparison of Response time after applying rules.

Table 5.7 Comparison of various performance prediction methods.

S. No.	Model	Inference	Percentage of Predictability achieved
1	Robocop	Prediction based on cost	65
2	Design Tuning Environment	Prediction based on distributed systems and dynamic rules	52.3
3	Rule engine	Prediction based on static rules	57.6
4	CLISSPE	Prediction based on expert systems, client server systems.	36.8
5	Proposed framework	Predictions based on availability, maintenance, reusability and rule based system	69.2

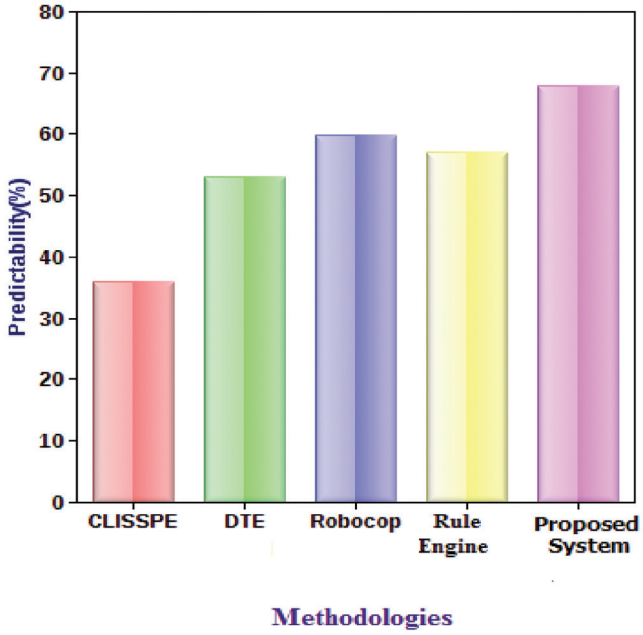

Figure 5.5 Comparison of various methods.

Table 5.8 Ranking of various attributes.

S. No	Attribute Name	Attribute	values	Cust req.	App/ feasi	Optimization level
1	Response time	1	0.35	0.3	1	9
2	Resource Utilization	4	86	95	5	7
3	Throughput	2	26	35	4	11
4	Timeliness	2	0.2	0.15	1	12
5	Maintainability	4	73	80	5	3
6	Reusability	5	60	70	2	8
7	Modularity	3	67	75	2	10
8	Affordability	5	50	60	5	4
9	Processor Utilization	3	95	95	3	5
10	Schedulability	5	70	75	5	1
11	Interoperability	5	60	70	5	6
12	Resource availability	4	100	100	4	2

5.9 SUMMARY

In this section, the effective of the proposed approach has been demonstrated. The chapter discusses the derived methodology with the analyzing and estimations of the proposed methodology. Here a mathematical formula is also derived to find out the probability of the failure system. It is felt that UML based approaches are better even after the standardization of queuing network. Addressing either qualitative or quantitative evaluation unique target formulism for the system assessment is used which means the software engineers are supported during the V&V activities. Performance indices are analyzed in this chapter i.e., utilization, throughput and response time according to iterations and system schedule. The forward path from software model to performance indices is represented by the modeling and analysis phases. Various approaches have been introduced towards model transformation and development of many performance model solvers. On the other hand, it can be noticed that there is a lack of automation and feedback models for elaborating and analyzing the results.