A Formal Approach using PEPA to Performance Analysis of Service-Oriented Architecture Style Specified through Graph Transformation System

Afshin Bamshadi
Department of software engineering, Faculty of computer engineering, Islamic Azad University, Malayer, Iran

Abstract

By spread of distributed extensive, concurrent software systems, and the necessity of possessing an efficiently acceptable software, it is required that performance evaluation be ensued during the preliminary processes of software development prior to implementation. Modifying the software following the implementation, with the purpose of increasing efficiency would be both cost-effective and time-consuming. Among the present architectural styles, service-oriented style is the best alternative to cope with extensiveness and distributedness due to its highest level of abstractiveness. Graph transformation system (GTS) is a formal, intelligible and dynamic language for architectural modeling. In this study, we have presented a method implying PEPA language for performance evaluation of service-oriented architectural style which has been modeled by graph transformation system. To assess performance evaluation through PEPA there is a need for identifying systems behavior and structure, which have been extracted from the graphs; this means that the architectural model specified by graph transformation systems has been transformed to PEPA performance model which is a formal modeling performance language based on process algebra. Finally action throughput of software systems and state of utilization of each component and also capacity utilization of each component has been analyzed and their related charts have been represented.

Keywords - *service-oriented architectural style, graph transformation system, PEPA, performance evaluation, software architecture.*

I. INTRODUCTION

In traditional approaches, performance analysis is conducted subsequent to implementation, which in case of low performance of software system, the designing operation must be repeated which is neither time-efficient nor cost-effective. In modern approach, which has been referred to as software performance engineering, performance analysis is predicted in the preliminary stages of development and prior to implementation [1]. In performance engineering, analysis is performed on architectural model the process in which the nuances are removed and only the whole are dealt with. The properties which are often examined in performance engineering include the following:

i) Throughput: the number of transactions which receive services in unit of time.
ii) Utilization: the fraction of time which the resource wastes while giving services to a customer relative to the overall time.

In architectural level, reflexes are made to components, their interaction and their relationships constraints [2]. Graph transformation system, as a formal, dynamic and highly expressive competency, can be used for architectural description. GTS can well express the structure and behavior of the system in which the nodes, components and edges display interactive and communicative behavior of the system.

In this study, service-oriented architectural style which have been modeled by graph transformation systems [3] have been employed as architectural model. The information required for creating performance model, i.e. system's structure and behavior, has been extracted and ultimately the PEPA model has been reached to. Then, adding the temporal information to PEPA model, performance evaluation has been conducted.

Among different architectural styles, service-oriented style, due to its high level of abstraction, is the best alternative for coping with complexity, extendedness and distributedness of today's systems to this end, performance evaluation has been conducted on this style. The concepts of object and component have some similarities with the concept of service; however, as can be seen in the Figure 1, the service-oriented style enjoys much abstraction level compared to that of object-oriented and component-based style.

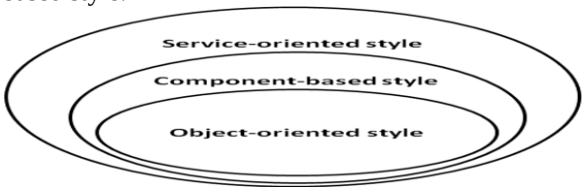

Fig.1 Comparing object-oriented, component-based and service-oriented styles from the viewpoint of level of abstraction

PEPA is a formal language based on process algebra and suitable for modeling and performance evaluation. There exist other methods for performance modeling including queuing network and petri net. However, process algebra has been employed in the present paper due to the followings reasons:

1. Formality: its mathematical base which can result in its accurateness, correctness and unambiguity.
2. Abstractness: removing the details and constructing a performance model by means of the system's generals, that is components and interactions.
3. Compositionality: constructing the system by means of interaction of collections of sub-systems which has simplified the model.

When performance model is created in PEPA, performance analysis is conducted through computing such characteristics as action throughput, states utilization of each component and also capacity utilization of each component. The proposed approach is present in Figure 2.

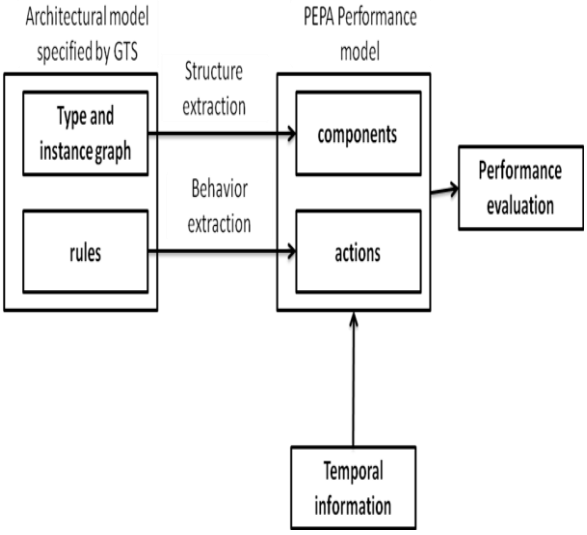

Fig.2 Proposed framework for modeling and performance analysis

The outline of the rest of the paper is as follows. In second part, the related works will be referred to. The third section will include PEPA language. The structural and behavioral elements of the style will be extracted in section four and five of the study respectively. The sixth section is devoted to transforming the proposed model to performance one. Performance evaluation will be dealt with in section seven and finally, conclusions and suggestions for further research will be represented in part eight of the study.

A. The study

To represent our proposed framework, the study of electronic travel agency presented in [3] has been used. This is represented in Figure 3.

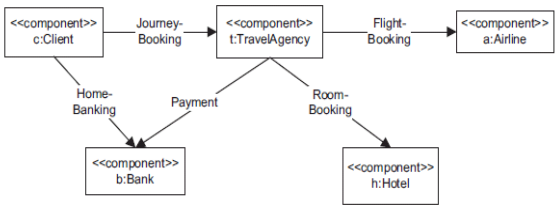

Fig.3 Components of electronic travel agency [4]

In Figure 3, each node displays one component and each edge shows one interaction or behavior. The sequence diagram related to the employed scenario in the study is displayed in Figure 4.

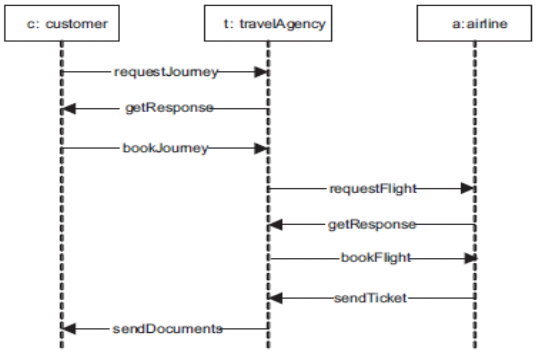

Fig.4 Sequence diagram used for the scenario of the study [4]

II. RELATED WORKS

In [5] to have fault tolerance system, first the core of service-oriented architectural style is developed, then different communication and reconfiguration for error tolerance have been developed by graph transformation rules and ultimately the proposed model using model checking technique has been verified for graph transformations systems.

In an approach to evaluate performance, Aquilani et al. [6] have presented architecture software. Their approach was the possibility of automatic ganging of a performance model based on queuing network (QN) from a dynamic architecture model based on labeled transition system (LTS).

In [7], from FDAF which is an aspect-oriented approach for designing and multi validate analyzing of non functional and real time properties has been used for transforming automatic architectural designing from UML to Rapid, and then performance analysis has been conducted on its security aspects.

Kloul [8] has presented an approach to modeling dynamic information and performance at the design level. This approach translates a UML 2.0 model into a processing algebra, PEPA nets. After building

the processing algebra model, the performance analysis can be done.

Baresi et.al [9], in an approach to architectural modeling and analyzing based on architecture style modeling, by means of class diagram accompanied by constraints and dynamic behavior using a graph transformation system has been presented. They have demonstrated their approach base on service-oriented style.

Security and performance in service-oriented applications has been investigated in [10] .in their research, they used genetic algorithm approach to find a collection of optimal services which can support commercial products.

Danil Kina and colleagues [11] have presented a simulation framework for evaluation restarting algorithm performance in service-oriented systems [SFERA]. They presented the SFERA framework for restarting simulation in service-oriented architecture.

In [12] modeling and verification for communicating reliable messaging has been used in service-oriented architecture systems. They first developed a core meta-model for service-oriented architecture with required parameters for communicating reliable messaging and then have modeled a reconfiguration for delivery reliable messaging by a graph model. In the end, a formal verification of a collection of proposed rules has been presented by means of combining analyzing tools required for transforming graph and labeled transition system.

In [13], an approach for automatic verification of graph transformation systems by Bogor has been presented. This approach supports both attributed typed graph and layered graph transformation system. The checked characteristics can be indicated directly either by linear temporal logic or transformation rules.

In [14] and [15], Heckel et al. have presented the idea of stochastic graph transformation system for evaluating non functional properties. Resorting to the fact that stochastic methods must be used for evaluating performance and reliability and also regarding the ability of graph transformation systems in creating dynamic models, they have presented such idea.

In [16] investigating and introducing AGG2.0 has been dealt with and several analysis technique have been introduced for graph transformation systems. AGG supports determining characteristic of algebraic graph transformation systems based on related characters.

III.PEPA(PERFORMANCE EVALUATION PROCESS ALGEBRA)

In this paper PEPA language is used for performance modeling. This language is based on process algebra. Due to its properties, this language is the best language for performance evaluation at the architectural level. The most important of which is the abstractness of language.

In PEPA, system is made of collective interaction of subsystems. Each activity in this language possesses an action type which is always abbreviated as type. Each activity of this language has a time period which is a stochastic variable with exponential distribution.

This parameter is called activity rate and is abbreviated as rate.

In PEPA language, an action is displayed as (α, r) where α is the type of action and r is a real number, the activity rate. Components and activities are primary and main parts of PEPA language. This language has several combinators which one can use in case of necessity combinators are presented in Table 1:

Table 1: PEPA Combinators

Subject	Writing form	description
prefix	(α, r).P	Display of sequencing behavior
collaboration	P <α, β> Q	Collaboration of P and Q components for α and β activities
choice	P + Q	Parallel processing in P and Q components

After specifying activities of each component using above combinators, the system equation must be created. The system equation is as follows:

Sys = (C1 <op1, op2> C2 <op3> C3)

The above example shows that C1 and C2 have interaction with op1 and op2 and C2 and C3 have interaction with op3 [17], [18].

IV.EXTRACTING STRUCTURAL ELEMENTS

In this section, the procedure for modeling related to structural section of service-oriented style, that is style elements and communicative constraints will be dealt with. Components for service-oriented style is consist of: component, connector, port, interface and operation. Component includes processes and system information. Component processes which are capable of be presented are accessible through interface. Communication among components is provided through connector. The internal structure of the component is hidden and can be accessed into only through its ports. Connector joins two interfaces to each other. Figure 5 demonstrates formal presenting of structural elements of service-oriented style and shows their connections by means of a graph scheme. In this graph, various nodes are equivalent for light elements and different edges, along with multiplicity of communication, display the topological constraints and specify how nodes are located beside each other in an architectural configuration. As an example, one connector can connects only two ports.

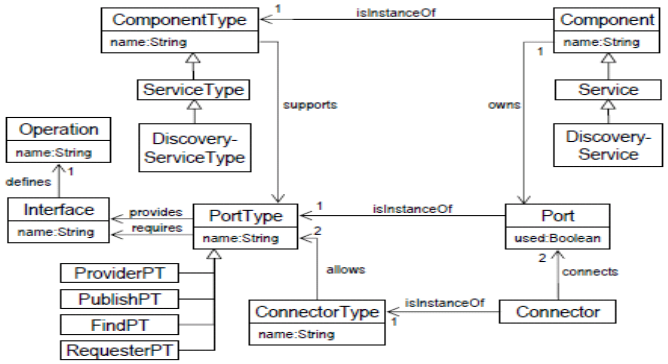

Fig.5 Structural section of graph scheme [3]

Architectural model which is based on the above style structures as an instance graph is located on graph scheme. Figure 6, is a part of architecture studied in this study.

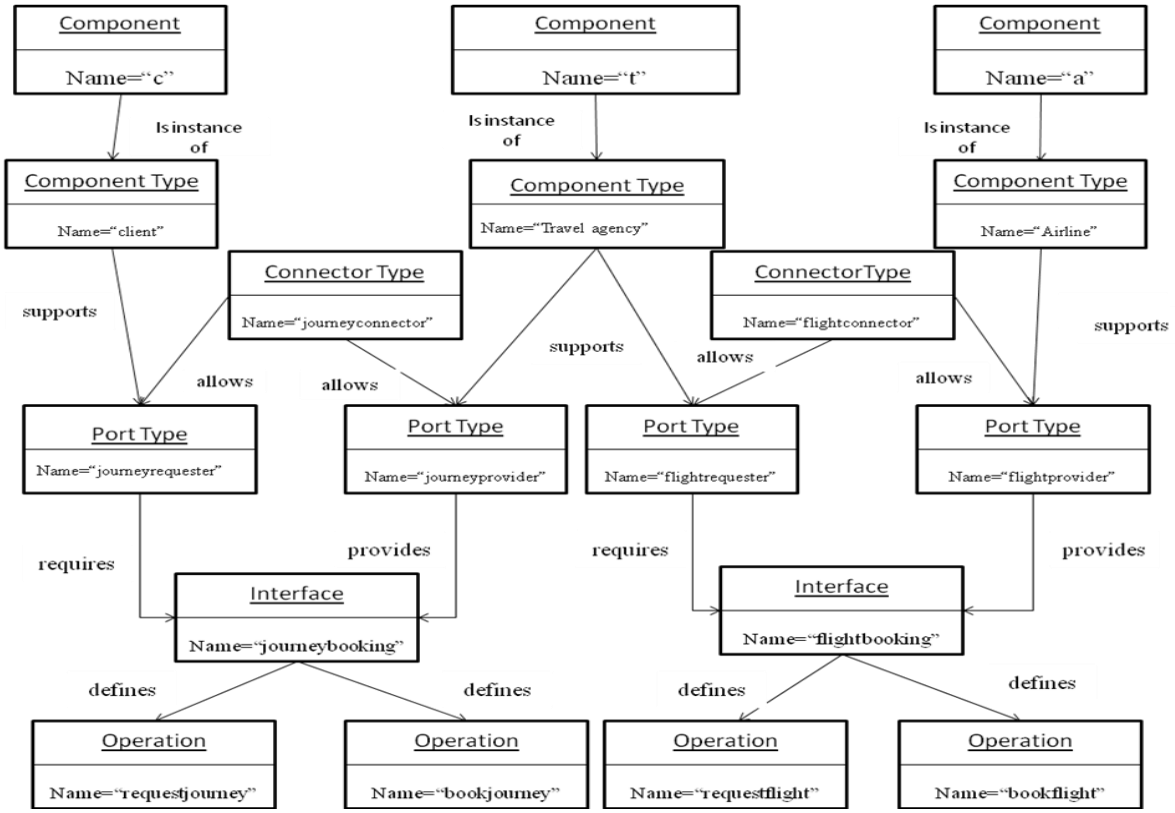

Fig.6 A part of instance graph under study [3]

In Figure 6, component types of Figure 5 which have depicted the general scheme of the study have been cut and their relevant operation of the description under study are extracted and connected to interface.

V. EXTRACTING BEHAVIORAL ELEMENTS

Style behavior can be regarded as all communicative mechanisms, since architectural components of service-oriented style in a constant and specific configuration have communication only through communicative mechanisms including connector, query sending, receiving query results so on.

In service-oriented style, complete list of communicative mechanisms and their description is as follows:

- Opening port: it is called in case of requirement for communicating with a component so that appropriate port is devoted to it.

- Closing port: when communication is disrupted, the port is released.
- Establishing communication: connecting communicative channel between two components.
- Communication disconnection: disconnection of communicative channel between two components.
- Call operation: sending call request related to one operation on one component by the operation requesting component.
- Receive call: receive call request for one operation on one component by operation providing component.
- Sending response: preparing and sending appropriate response for a requested operation from operation provider component to requestor component.
- Receive response: receiving response of operation request on one operation requesting component.
- Send service publication: service provider of a message sends the service publication to service discovery.
- Receive service publication: the message of service publication sent by service provider is received by service discovery.
- Send service query: a service query message is sent to service discovery on behalf of service requester component.
- Receive service query: service discovery receives the message of service query from the service requester.
- Send query result: response message for query result is send from service discovery to service requester.
- Receive query result: service requester receives the message which is result of the query.

The collection of above behaviors, which are termed as required behavioral mechanisms, cover all existing behavior requests between components in this style. Each of above behaviors is expressed through a rule, for further study of it, can refer to [3].

In graph scheme of Figure 7, the nodes request, response, service publication, query result and service query show different messages, the common property of those messages is in message node. A message is sent through a sending port and received through a connector.

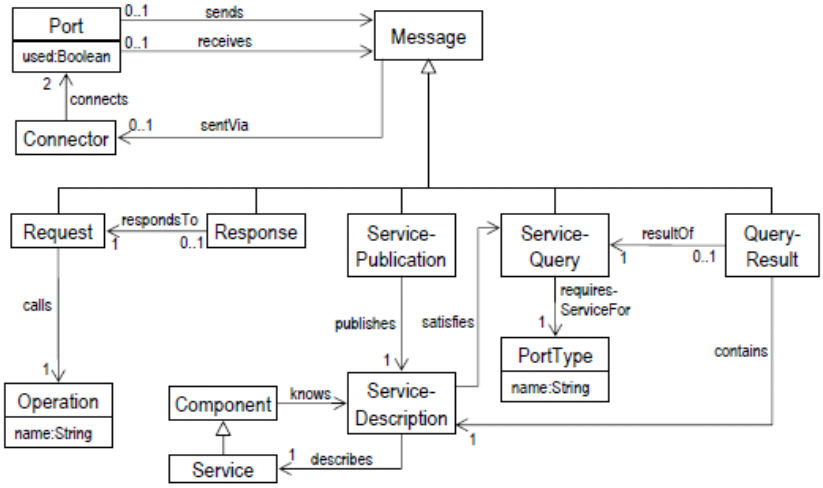

Fig.7 Graph schema containing structural and behavioral components [3]

VI. TRANSFORMING THE PROPOSED MODEL INTO A PERFORMANCE MODEL

Main elements of PEPA language includes: components and operations, in other words, other activities and their combinations inside the system for achieving a general system and its evaluation. Its transforming algorithm includes the two following phases:

- Identifying and extracting structural elements of performance model.
- Extracting interactive behavior of components and creating system equation.

A. Identifying and extracting structural components of performance model

As we know, components forming PEPA language include components are variety of activities.

These elements in graph scheme (i.e. component and component type) are equal to the concepts of component type in performance model so, to identify them, searching must start from start

graph and for each component type, the amount of name property in it is added to the list of PEPA model component.

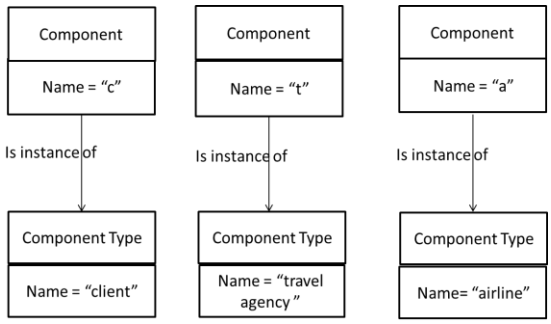

Fig.8 Start graph for the case under study [3]

Since, to achieve the general equation of PEPA, knowing the number of participating components in system is required, in investigating start graph for model, one can, for each component type, consider the number of node components of its type as the number of its corresponding components.

As an example, in Figure 8 a part of model start graph under study for three components (Travel agency, Client, Airline) is displayed and for each of these components, there is one sample of participating components.

The set of activities related to component consists of a set of defined operations on component type which are equivalent to what can be achieved. After achieving these collections, its segregate as a non-repeated set, forms all activities related to a model.

B. Extracting interactive behavior of components and creating system equation

In this section, first we extract interactive behavior and after that, we create the system equation by PEPA language. Based on information extracted from graphs, interactive behavior of the system is as follows:

Customer requests journey from travel agency

Customer = (requestjourney, r).Travel Agency

Travel agency responds to the customer request

Travel Agency = (tsendresponsetoc, r). Customer

Customer sends the booking journey request to travel agency

Customer = (csendbookjourneytot, r). Travel Agency

Travel agency requests for flying from airline

Travel Agency = (requestflight, r).Airline

Airline responds to travel agency's request

Airline = (asendresponsetot, r). Travel Agency

Travel agency sends book flight to airline

Travel Agency = (bookflight, r). Airline

Airline books the flight and sends the ticket to travel agency

Airline = (sendticket, r). Travel Agency

Travel agency sends flight documents to the customer

Travel Agency = (tsenddocumenttoc, r). Customer

The system equation is as follow:

((Customer < requestjourney, tsendresponsetoc, csendbookjourneytot, tsenddocumenttoc> Travel Agency <requestflight, asendresponsetot, bookflight, sendticket > Airline))

VII. PERFORMANCE EVALUATION

In this section, our proposed model for case study of electronic travel agency will be evaluated. In Figure 9, system throughput chart can be observed for different actions, in which throughput for eight main actions (other than required actions for communication) is the same which is an indication of balanced behavior of service-oriented style in relation to its actions. The greatest amount of throughput is related to opening and closing port components of travel agency before and after communication which makes it possible to have quick communication with this component.

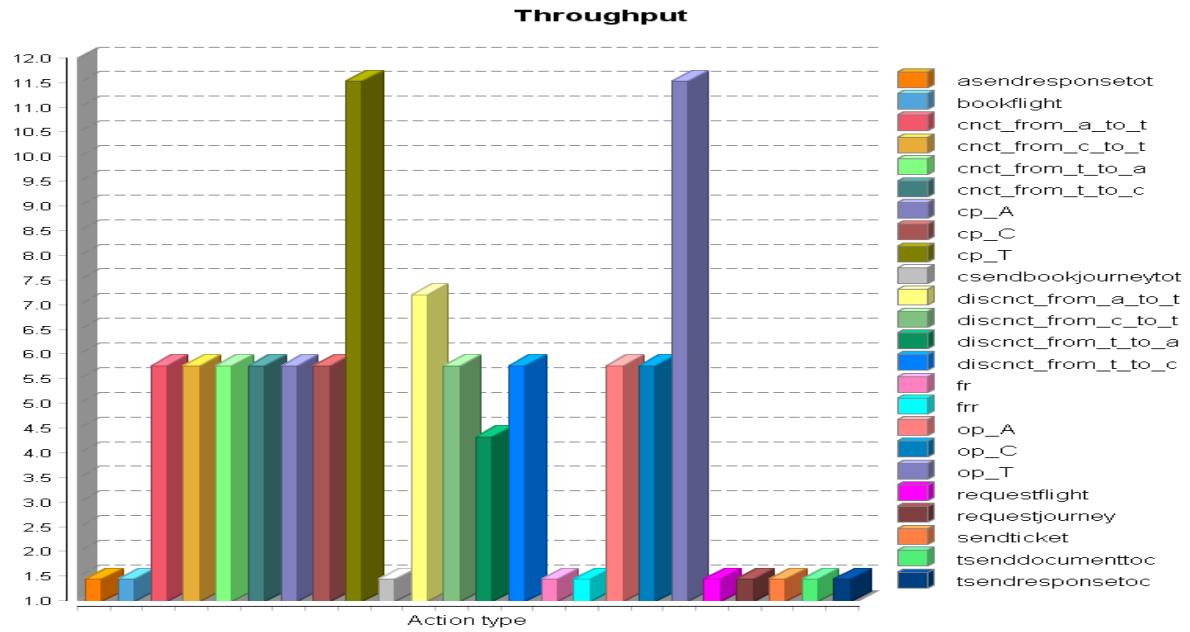

Fig.9 System throughput for different actions

Figure 10 shows state utilization of customer component in which the highest rate of utilization belongs to the state of sending flight documents by travel agency to customers.

Due to high rate of this state utilization, system performance is very high in viewpoint of customers and they are satisfied with the system.

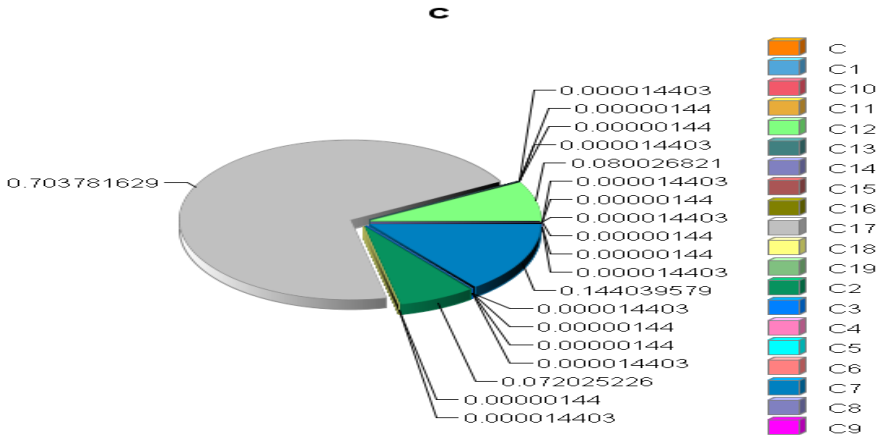

Fig.10 States utilization of customer component

In Figure 11, state utilization of the travel agency component can be observed in which the highest level of utilization belongs to the state of sending flight request response from airline to travel agency.

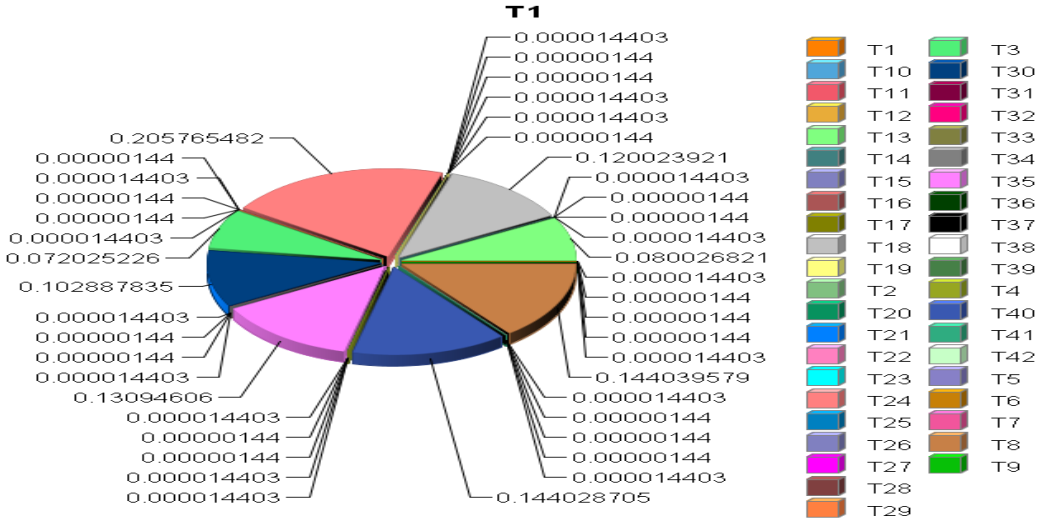

Fig.11 States utilization of travel agency component

In Figure 12, state utilization of airline component can be observed in which the highest rate of utilization belongs to receiving flight request by airline from travel agency.

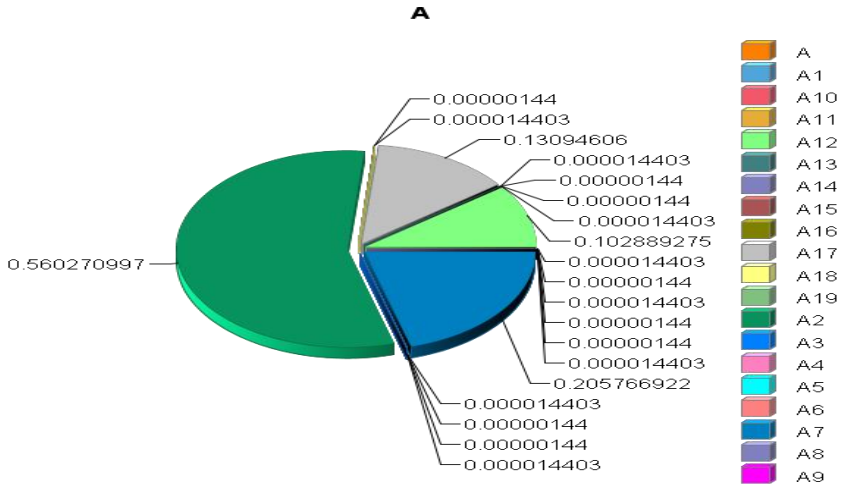

Fig.12 States utilization of airline component

VIII.CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

What we followed in this study was, presenting a method for performance evaluation of service-oriented architectural style which has been modeled by graph transformation system by PEPA language.

First, modeling pattern for service-oriented architectural style by graph transformation system has been introduced. Next, and algorithm has been introduced which can transfer the present model to the one with which one can perform performance analysis, this in fact transforms the architectural into a performance one.

In this study graph transformation system was used to describe architecture and PEPA was used for performance modeling and the two methods are formal and unambiguous.

Based on charts resulted from PEPA in this study, one can conclude that in service-oriented style:

1. Main actions of this style (other than actions used for communication) have the same throughput and any system based on this style has a balanced throughput.
2. Granting services to customers is followed with good utilization; therefore customers are satisfied with system performance.
3. Capacity utilizations of different components are much similar to each other and it acts normally from viewpoint of utilization.

Future works in the area of graph transformation systems can be such as modeling of other styles such as layered styles, client/server, pipe and filter and blackboard style by graph transformation system could be focus of concern. In the field of performance analysis, one can investigate the perspective present in this study on other styles.

REFERENCES

[1] T. Kauppi, "Performance analysis at the software architectural level," Technical report, ISSN: 14550849, 2003.

[2] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, Software Architecture: Foundations, Theory, and Practice, New York, NY, USA: John Wiley, 2008.

[3] S. Thöne, "Dynamic Software Architectures A Style-Based Modeling and Refinement Technique with Graph Transformations," Ph.D thesis, University of Paderborn, Paderborn, Germany, 2005.

[4] V. Rafe, "Senario-driven analysis of systems specified through graph transformation," Visual Language and Computing, vol. 24, pp. 136-145, 2013.

[5] V. Rafe, and F. Mahdian, "Style-based modeling and verification of fault tolerance service oriented architectures," Procedia Computer Science, vol. 3, pp. 972-976, 2011.

[6] F. Aquilani, S. Balsamo, and P. Inverardi, "Performance analysis at the software architectural design level," Performance Evaluation, vol. 45, pp. 147-178, 2001.

[7] L. Dai and K. Cooperb, "Modeling and performance analysis for security aspects," Science of Computer Programming, vol. 61, pp. 58-71, 2006.

[8] L. Kloul, "Performance Analysis of a Software Retrieval Service," Electronic Notes in Theoretical Computer Science, vol. 232, pp. 145-163, 2009.

[9] L. baresi, R. heckel, S. Thone, and D. Varro, "Modeling and Validation of Service-Oriented Architectures," Application vs. Style. Proc.3th Int. Conf. ESEC/FSE. pp. 68-77, 2003.

[10] H. Zo, D. L. Nazareth, and H. K. Jain, "Security and performance in service-oriented applications: Trading off competing objectives," Decision Support Systems, vol. 50, pp. 336-346, 2010.

[11] A. Danilkina, P. Reinecke, and K. Wolter, "SFERA: A Simulation Framework for the Performance Evaluation of Restart Algorithms in Service-Oriented Systems," Electronic Notes in Theoretical Computer Science. Vol. 291, pp. 3-14, 2013.

[12] L. Gonczy, M. Kovacs, and D. Varro, "Modeling and Verification of Reliable Messaging by Graph Transformation Systems," Electronic Notes in Theoretical Computer Science, vol. 175, pp. 37-50, 2007.

[13] L. Baresi, V. Rafe, A. T. Rahmani, and P. Spoletini, "An Efficient Solution for Model Checking Graph Transformation Systems," Electronic Notes in Theoretical Computer Science, vol. 213, pp. 3-21, 2008.

[14] R. Heckel, G. Lajios, and S. Menge, "Graph Transformation, " Lecture Notes in Computer Science,; vol. 3256, pp. 210-255, 2004.

[15] R. Heckel, "Stochastic Analysis of Graph Transformation Systems: A Case Study in P2P Networks," Lecture Notes in Computer Science, vol. 3722, pp. 53-69, 2005.

[16] O. Runge, C. Ermel, and G Taentzer, "AGG 2.0– New Features for Specifying and Analyzing Algebraic Graph Transformations," LNCS, vol. 7233, pp. 81-88, 2012.

[17] J. Hillston, " Tuning Systems :From Composition to Performance," The Computer Journal, vol. 48, pp. 385-400, 2005.

[18] J. Hillston, "A Compositional Approach to Performance Modeling," Ph.D thesis, University of Edinburgh, Edinburgh, Scotland, 1994.