IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 6, Dec-Jan, 2014
ISSN: 2320 - 8791
www.ijreat.org

PEVOL: A Method for Annotating and Analyzing the Software Design

R. Aroul canessane[1], Dr. S. Srinivasan[2]

[1]Research Scholar, Sathyabama University, Chennai.

[2]Professor & Head, Dept. Of Computer Science and Engineering, Anna University, Madurai

ABSTRACT : A variety of approaches have been introduced for creating a performance models for software. At a specific point of the software lifecycle, to evaluate and develop the presentation of software system under development is still being a research issue. Here we have proposed a novel scheme in order to develop a performance model by integrating the software and hardware with the help of the performance indices. We have implemented with the help of a visualizing software design tool, which makes transformation of UML to queuing networks and the analysis is done using a probabilistic rule based algorithm which has been implemented using java.

Keywords– Performance, transformation, architecture, connectors, feedback.

INTRODUCTION

The functional and nonfunctional characteristics of software can be analyzed at the design stage of the software development. There are many semiformal methods and notations used in this regards, such as petrinets, qualitative logics, automata theory and calculus which can be generically used for UML, which some of them are still in research. The assessment of the characteristics can be analyzed at the design stage by using alternative architectural design which solves the functional and non functional characterizes. Performance and their constraints are the major influential facts that support architectural design choices. A method has been proposed for improving and predicting the software architecture performance. The methodology PEVOL has many phases where the performance indices can be assessed for the different scenarios of software architecture at design level.

On the basis of the indices, it helps to decide to improve the design or discard the design before implementing. Thou PEVOL is independent from standard notations of architectural design we focus on visualization design tool which gives a description based on the functional verification by model checking and performance evaluation through PEVOL a rule based model. On the analysis side PEVOL employs the queuing network. The main aim of choosing queuing network is, it supports the relation between the elements and components of the architectural description. The performance indices which we have analyzed are utilization and maintainability. The transformation of visualization tool to queuing network implies a major set of performance analysis technique on the architectures. This paper presents a part of implementation of PEVOL and is organized as follows. Section 2, UML Visualization tool towards queuing networks. Section 3, queuing networks. Section 4, the Architecture. Section 5, Implications & Results and Section 6, Conclusion.

I. VSDT TO QUEUING NETWORKS

A transformation towards queuing networks has been implemented with the help of VSDT, the idea of transformation process is done with the help of QNBE (Queuing network basic elements). The transformation is detailed hierarchically which has helped us a lot in our implementation of PEVOL. The hierarchy is the collection of action, behavior, classification and communication pattern which is a combination of QNBE.

a. VSDT

Visualization software design tool is a description language of software architecture based on the process algebra. The VSDT represents the architectural type which shares the behavior and topology of the component as in Table1. The description of VSDT starts with the name and it has various parameters of initialization and default values.

```
Design type
      Behavior
        Behavior elements……..
        Input / output
        connectors………
      Topology
        Instances……….
        Interactions…………
        CRC……….
End
```

Table 1. Behaviour and Topology of Components

The VSDT shows the complete behavior of the software component and the relations using the connectors. The sequence behavior shows either the process to stop or continue. The duration of each process element is calculated using the stochastic process which is defined in three categories as passive, immediate and continue. The interactions that are made using the behavioral diagrams are classified from various input and output interactions.The qualifier communication shows the multiplicity of communications that has interaction with it, the parameters and connectors are shown under this qualifier. The communicating must be noted if and only if it has an interaction with other components. Using the state transition graph, the stochastic process can be evolved using the probabilistic theory.

b. QUEUEING NETWORKS

Queuing Networks is a combination of different interactive centers of services which are represented for the sharing of resources by the customer classes, queuing networks are the structured models used for the performance of the components and their connection. This is an advantage for the design phase of software architectures. The average performance is calculated for the performance indices such as utilization, throughput, workload etc. at the entire queuing network service centers. The diagnostic information can be got for the components, connectors, interaction and behaviors. At the later stage an algorithm can be proposed for solving those diagnostic information's.

Using the queuing networks we can solve the solutions for the service centers in respect of individual solutions and we can integrate that solution later using multiplicity which gives a major support in performance analysis for the components used in architectural descriptions. Finally the solutions can be given using the queuing network solver, thou we don't know the actual system performance indices at the design level, we can use this feature in design level. The various QNBEs are shown in fig1. Where *i* denotes the various destinations, *g* denoted for the customer number classes, the arrival and the time required for the services are done using the phase distribution.

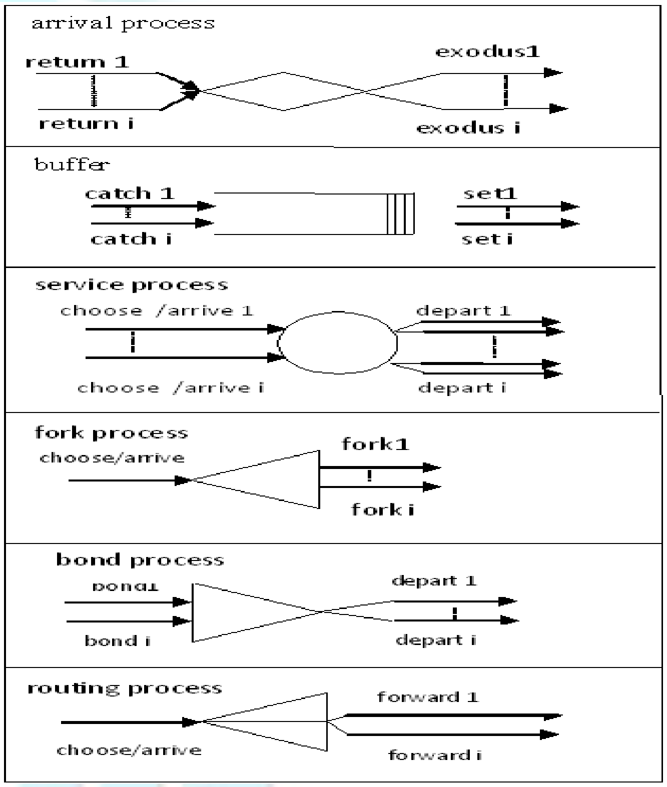


Fig1. Basic elements of Queueing elements

The arrival of customer classes are generated, for a single and multiple arrivals separate unbounded population instances are created. In case of finite class an explicit model is created for the customer. A buffer is created for the waiting queue to get a service at an instance based on the round robin method. Bounded buffer is used if any class is not extended. A service process server is used for the customer's different classes. A fork is used for the different child request process to the appropriate server classes. A routing process forwards the customers of the classes to different destinations with respect to the appropriated connection that has to be made using the connectors. The approach that we have made is to transformation mapping between the VSDT to the QNEB as shown in the fig1. We have followed a starting from the scratch towards the growth of the implementation which uses the basic elements of QNEBs. The different QNBEs that we have used in this approach are Actions and Behavioral patterns in Table. 2. and their corresponding QNBEs.

Actions/ Behaviours	**Actions/ Patterns**	**QNBE**
Input	return/ return get/ with and without condition select/ select visit/ visit	Single client on wait Buffer capacity utilized Fork process Unbuffered process
Output	exit / exit put/ put	Buffered process which uses fork, arrive and routing Buffered process
Internal	Phase/ exit fork/ join	Single process with arrival, fork and routing Fork and Join process

Table 2. Actions and Behavioural Patterns

c. BEHAVIOURAL PATTERN RULES WITH RESPECT TO QNEB

The rules that are depicted in Table 2 are the same for the fig1. The rows of Table2. Represents the QNBEs and the combination of the actions and behavior pattern rules are defined in the second Colum, the additional assumptions are considered which has to be verified before generating the QNBEs. We have certain rules for depicting the QNBEs multiple arrivals, infinite arrival, buffered fork and join process.

II. VSDT TO QUEUING NETWORKS

The PEVOL model is enabled using the VSDT to Queuing Network a java tool for transformation of VSDT to queuing networks. The topology which we have used has some syntactic restrictions which are complemented by QNEB. The QNEB is connected to create a formalized Queuing Networks with respect to the rules. The instance of the behavioral pattern is shown in table 2 with the help of QNEB instances. The Queuing Networks which has been used stores the performance information, where the VSDT is used as an interchange format which makes the enough input for the Queuing networks.

III. THE ARCHITECTURE

We have used architecture for the evaluation of verification of functions and non functions, performance evaluation and analysis in turn creates an optimal method for the evaluation of performance indices using probabilistic theory. The architecture shown in the fig2. Shows how performance evaluation is made with respect to the queuing networks. The typical architecture shows the UML design to VSDT, which provides an input for queuing networks.

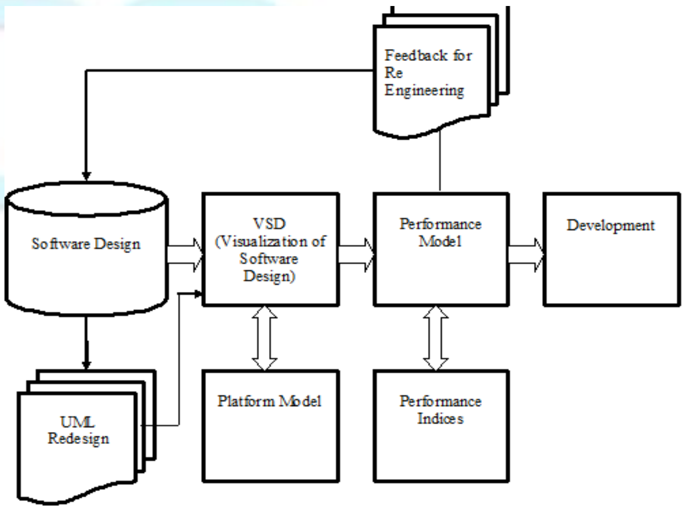
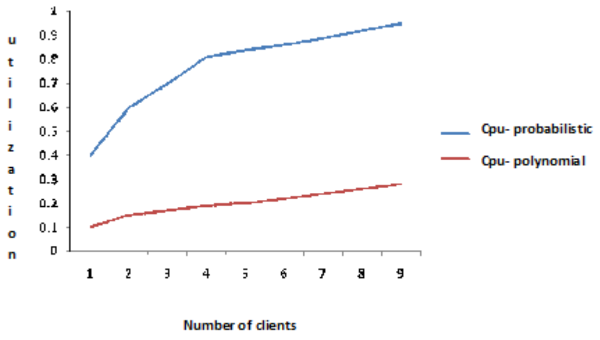


Fig 2. The Architecture

The output of the queuing network is evaluated using the performance indices. If the indices are satisfied we go for the development else we give a feedback for changing/ modifying the design to get the desired output to get better software. We have developed a queuing network solver using the rules which can provide a graphical output developed in java environment. The solutions are given and represented using a graph.

IV. IMPLICATIONS AND RESULTS

This section show an implication for the PEVOL on the Design developed for the ATM, a system made of getting an input through the hardware and performs the operation with the help of the software in a distributed system. The common services that are requested by the hardware are making the translation such as withdraw, deposit and additional options on the banking transaction. The ATM hardware performs these operations and received by a distributed system. The experiments illustrate the usage of the hardware, validating the transactions (using the VSTD to queuing networks) and using probabilistic theory checks the performance using the PEVOL for utilization and maintainability. The implementation is not completely shown here since we have some constraints to show it and partly under development. The utilization of CPU, Video and Memory have a service times of 0.5ms, 1ms and 0.5ms respectively, the values that are shown are approximate values of the scenarios. The parameter that we have chosen for the performance analysis is the number of users N, and the time for the operations as T for each transaction. Some of the direct measures are listed below.



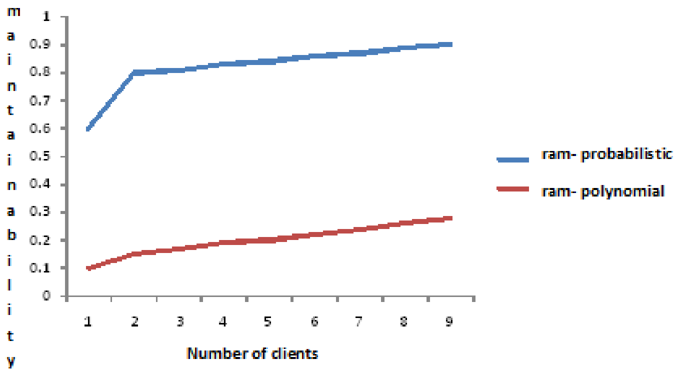


Fig3. Utilization and Maintainability

In fig3. We have shown the utilization and maintainability that we have got for the ATM. We have applied a probabilistic theory for the analysis that is shown graphically and it compared with an existing model, which uses a queuing network in a polynomial based algorithm. The actions that we have taken in our application for this implementation are shown in the Table 3 below with respect to the behavioral of QNBE's.

Table 3. actions considered in the ATM

Action	Service Time (ms)	Mapping
1	0.5	Phase
2	0.25	Branch
3	0.25	Phase
4	0.3	Phase
5	0.45	Logical
6	0.5	Logical
7	0.75	Logical
8	0.8	Hardware
9	0.2	Phase

VI CONCLUSION

In this paper we have used the VSDT for solving the queuing network models which supports the PEVOL for the ATM. Thou our method try to exploit the queuing network, it tries to solve the evaluation process for larger architectures in respect of queuing network solvers. Many approaches are there for transformation of architectural models to a performance model, but all of them are not implemented as working tools. Since we have used an UML as an architectural description language as source notation, the future work can be moved using an architectural description language. We would also like to compare with those methods without exploiting the architectural framework.

References

[1] Kai Huang , Wolfgang Haid , Iuliana Bacivarov , Matthias Keller , Lothar Thiele, Embedding formal performance analysis into the design cycle of MPSoCs for real-time streaming applications, ACM dl., Volume 11 Issue 1, Pp 211 – 220, March 2012 .
[2] André van Hoorn, Et.al, Kieker: a framework for application performance monitoring and dynamic software analysis, Proceeding ICPE '12 , ACM dl.,Pp 247-248, 2012.
[3] Christina J. Hopfea,Jan L.M. Hensen,Uncertainty analysis in building performance simulation for design support, Volume 43, Issue 10, Pp 2798–2805, October 2011.
[4] Marco Bozzano, Et. Al., Dependability and Performance Analysis of Extended AADL Models, Computer Journal, Volume 54, Issue 5,Pp. 754-775,2011.
[5] Vitaly Chipounov, Et.al, S2E: a platform for in-vivo multi-path analysis of software systems, Proceeding ASPLOS XVI, Pp 265-278, 2011.
[6] Anne Martens, Heiko Koziolek, Steffen Becker, Ralf Reussner ,Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms, Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering,Pp 105-116 ,2010
[7] Remco R. Bouckaert,Eibe Frank,Et.al,WEKA:Experiences with a Java Open-Source Project,The Journal of Machine Learning, Volume 11,Pp 2533-2541,2010.
[8] Jing Xu,Rule-based automatic software performance diagnosis and improvement, Elsevier, Volume 67, Issue 8, Pp 585–611, August 2010.
[9] Heiko Koziolek,Performance evaluation of component-based software systems: A survey,Elsevier, Volume 67, Issue 8, Pp 634–658, August 2010.
[10] R. Aroul canessane, S. Srinivasan, A Study of SSASS: Methods for Analyzing the Software Architecture, CIIt, Vol5,No6,pp 212 - 217,2013.
[11] R. Aroul canessane, S. Srinivasan, UML Model Transformation for a Product Line Design,IJET, Vol 5,No5, pp 3726-3733, 2013.
[12] Reza Vaziri, Reza Javidan,A Bayesian Based Model for Evaluation of Software Architecture Reliability, European Journal of Scientific Research,Vol.63 No.2, pp.243-254,2011.
[13] Cheung, L., Roshandel, R., Medvidovic, N, Early prediction of software component reliability, Proceedings of the 30th international conference on Software engineering, Leipzig, Germany, 111-120, 2008.
[14] Hendrickson,Et.al., Modeling Product Line Architectures through Change Sets and Relationships. In Proceedings of the 29th international Conference on Software Engineering. International Conference on Software Engineering. IEEE Computer Society,washington, DC, pp. 189-19, 2007.
[15]Wagner, Et. Al., Modeling SoftwarMachines: A Practical Approach, by Taylor & Francis Group, LLC, 2007.

R. Aroulcanessane received the M.E.in Computer Science and Engineering from Sathyabama University, Chennai , Masters of Computer Applications from St. Joseph’s College of Engineering, Chennai, University of Madras. He is presently pursuing the Ph.D degree in the Department of Computer Science and Engineering, Sathyabama University, Chennai, India. He has 13 years of experience in Teaching. He has also held various responsibilities as a part of his research. He has published around 8 research papers in journals and conferences. He has written books for some of the universities. His research area is Software Engineering and also interested in Data Base Management Systems and Data Warehousing and Mining.

S. Srinivasan received the Ph.D degree in Computer Science & Engineering from the Sathyabama University, Chennai, and M.Tech. in Computer Science & Engineering from the Indian Institute of Technology, Chennai, and M.B.A. in Systems & Finance from Sathyabama Engineering College, University of Madras, Chennai, and the M.Sc. in Mathematics from the Gobi Arts College, Gobichettipalayam, Bharathiar University, Coimbatore. He is presently working as Professor and Head, Department of Computer Science and Engineering , Anna University, Regional Centre, Madurai. He has 20 Years of experience in Teaching and Research. He has also held various positions and responsibilities in Technical Institutions. He is acting as expert member at various universities in various capacities.. He has published more than 40 research papers in journals, books, conferences, and workshops. His research interest includes text mining, data mining & data warehouse, and Software Engineering.