International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.5, Sep-Oct. 2012 pp-3402-3405 ISSN: 2249-6645

Analysis of Design Level Defects Based On PetriNet Model

P. Rajesh[1], P. Rajarajeswari[2], Dr. D. Vasumathi[3]

** (M.Tech, Computer Science and Engineering, Madanapalle Institute of Technology & Sciences, Madanapalle, A.P,India)*
*** (M.Tech[Ph.d], Assistant Professor, Department of CSE, Madanapalle Institute of Technology & Sciences, Madanapalle, A.P, India)*
**** (M.Tech, Ph.d, Associate Professor, Department of CSE, JNTU Hyderabad, Hyderabad, A.P,India)*

***Abstract*:** *Software Designs must be evaluated in Software Development Process so as to avoid bugs and unsatisfactory performances. Concerning performance and real-time properties modeling, OMG specified the UML Profile for Schedulability, Performance and Time Specification (SPT). UML is the standard OO modeling Language our system, but UML is too static to model the performance.It is not able to capture dynamic nature of system. So here we are using PetriNets to capture Dynamic nature of the System for UML use Case Diagrams and collaboration diagrams. First we draw the UML Use Case diagrams and collaboration diagrams with SPT (i.e. performance information) and then convert them into Executable Petri Net models. Finally we consider a case study for our proposed algorithm.*

***Keywords*:** *Software Performance Engineering, UML, Petri Nets, Performance Evaluation.*

I. INTRODUCTION

Performance is an important but often overlooked aspect of the software design. Indeed, the consideration on performance issues is in many cases left until late in the software development process (SDP), when problems have already manifested themselves at system test or within the deployed system. The identification of possible bugs or unsatisfactory performance in the design phase allows to contain the costs, also permitting to compare different alternatives. This kind of approach implements so called software performance engineering (SPE)[2],which is a systematic, quantitative technique to construct software systems that meet performance objectives.

Using the UML for modeling and the OMG UML Profile for Schedulability, Performance and Time Specification (SPT)[9] to specify performance requirements into a UML model therefore mapped into a performance model(Petri Nets). In this perspective, the main contribution of this paper is the implementation of a software development process which takes into account performance specifications and requirements: the software performance engineering development process (SPEDP). SPEDP includes and synthesizes both the aim of modeling and developing generic software architecture, and the aim of investigating the performance of the overall (hardware/software) elaboration system. It can be applied both in early phases, as a software performance engineering technique, and in test phases, as a common software performance evaluation technique. SPEDP fixes steps, rules and guidelines to follow in order to achieve the desired results in the software development satisfying the performance requirements.

UML is too static to model dynamic behavior of the systems. So to overcome this we use Petri Nets as to develop Executable models.

Bernardi et al. have proposed the automatic translation of state charts and Sequence Diagrams into Generalized Stochastic Petri Nets, as well as a composition of the resulting net models suitable for reaching a given analysis goal [3]. Elkoutbi et al. have transformed a simple use case structure to colored Petri nets [4] and Kamandi et al. have transformed use case to Object Stochastic Activity Network (OSAN) [5]. Different approaches are used for the transformation of sequence diagrams to Petri nets. In the approach proposed by Bernardi et al., all structures of the sequence diagram have been transformed to Generalized Stochastic Petri Nets [3]. Ourdani et al. transformed the simplest structures in the sequence diagram to colored Petri nets [6]. The difference between the two transformations is that in Bernardi et al.'s approach [3] the transformation is based on mapping messages as well as conveying them, while in Ourdani et al.'s approach [6] the transformation is based on message sender and receiver component.

Although so many researchers have used this Petri Nets as performance Domain, none of the Researchers utilized the collaboration diagrams and Use Case diagrams for performance evaluation [3, 5] of software design description based on Petri Nets either in this paper we propose an algorithm for transforming annotated collaboration diagrams to Petri Nets by referring [1].

While Use Case diagrams model the functions of system components Collaboration diagrams model the interaction between the system components (i.e. the messages exchanged between the components).These diagrams enriched by performance input parameters. Then we must transform these input parameters to tokens of places or guards for arcs and transitions in the Petri Net Model .After that target model must be evaluated. The designers will decide whether and how software architecture should be refined from analysis of the results of the analysis of the results of the evaluation steps.

The rest of the paper is organized as follows. In section 2 an algorithm for transmission of the collaboration diagrams and Use Case enriched by performance parameters will be proposed. Section3 presents a case study which illustrates the proposed algorithm. Ultimately the section 4 concludes the paper.

II. THE TRANSFORMATION OF UML DIAGRAMS TO PETRI NETS

In the following, the collaboration diagrams and role of them concerning performance will be explained. Also we use UML Profile for Schedulability, Performance and Time (SPT) to annotate the additional information to these diagrams. Then the detail of the transformation algorithm will be explained.

A. Role of the collaboration Diagram concerning performance

Collaboration diagrams are useful design tools because they provide a dynamic view of the system behavior, which can be difficult to extract from static diagrams or specifications. To each message in the diagram a condition can be attached, representing the possibility that the message could be dispatched. Even multiple messages can leave a single point each one labeled by a condition. From the performance point of view it can be considered that routing rates are attached to the messages. We use PAprob tag value to give such information. A set of messages can be dispatched multiple times if they are enclosed and marked as an iteration. This construction also has its implications from the performance point of view [7]. We use the PAprob tag value of the PAstep stereotype to annotate the probability of execution an alternative behavior when the sequence diagram presents either alt, break or option fragment operator. Also iteration will be represented by the tag value PArep [7]. The PAdemand tag value specifies the duration of the activities as random variables exponentially distributed, which are the ones supported by the extension of Petri nets formalism.

Also PAdelay tag value specifies the delay of the messages exchanged among components allocated in different physical nodes [7].

B. *The Transformation of Collaboration diagrams to Petri Nets*

-Petri nets representation of asynchronous and delayed messages: We use PAdemand tag value to give such information. In this state, the client component is displayed in the form of place-transition-place. But the server component is shown as place-transition-place-transition- place. The second transition is a timed transition with an assigned firing rate. Figure 2.a shows this type of messages.

Petri nets representation of synchronous and delayed messages: We use PAdemand tag value to give such information. In this state the client component is shown in the form of place-transition-place-transition-place and their connection can happens through two shared places. But the server component is displayed as place-transition-place-transition-place-transition-place where central transition is of the timed transition. Figure 2.b shows this type of message.

Figure 2.c depicts the message exchanged between two annotated components c1 and c2 with PAdelay tag value and the resulting Petri net, where t1 represents the sending action performed by component c1, t2 models the message transmission delay and t3 represents the reception of the message by component c2. The value associated to the tag PAdelay defines the firing rate of the timed transition, t2.

Figure 3 shows the two types of sequence diagram constructors (alternative and loop) and their mapping onto Petri nets. The translation of these constructors requires the use of additional Petri net sub-nets. Figure 3.a2 shows the Petri net sub-net modeling, the alternative choice between ev1 and ev2. The additional sub-nets are enclosed in the dotted rectangle. Figure 3.b2 shows the Petri net modeling as an optional choice. Consider that the choices in these two figures are probabilistic. The weights of the conflicting transitions t1 and t2 are derived from the tag value PAprob attached to the constraint condition. Finally, Figure 3.c2 and 3.c3 demonstrate the Petri net sub-net modeling, a while-do loop and repeat-until, respectively. The sub-nets repeat and while models the iteration of message ev1.

C. *Role of the Use case Diagram Concerning Performance*

Use Case Diagram describes the software system at a Very high level of abstraction by identifying its functionalities. Thus this type of diagram gives information on the type of traffics incoming in the system. The use case diagram should represent a Performance Context, since it specifies one or more scenarios that are used to explore various dynamic situations involving a specific set of resources. Then, it is stereotyped as PAcontext. Each use case used with performance evaluation purposes could represent a step. Then, they are stereotyped as PAstep. The performance annotations for the use case diagram are the assignment of a probability to every edge that links a type of actor to a use case, i.e. the probability of the actor to execute the use case [7, 8]. We use PAprob tag value for this annotation. Also each use case of interest should be detailed by means of a Collaboration diagram.

D. The transformation of Use case Diagrams to Petri Nets

In this paper, in order to transform a use case diagram to a Petri net, the idea presented by Elkoutbi et al. [4] is applied with some adjustment. The annotated use case diagram can be transformed to the Petri net through the following steps:

The transformation of each use case to a Petri net: In this transformation, each use case is transformed into a Petri Net model. We use one place for each actor and one dark place for each use case. The input for Place is a transition with a guard. In the next stage, the dark places are replaced with the obtained Petri net from the Collaboration diagram. Figure 1 displays an actor and two use cases that are annotated with SPT profiles. Then these diagrams are transformed to equivalent Petri net. The selection condition of each use case is assigned to t1and t2 transitions.

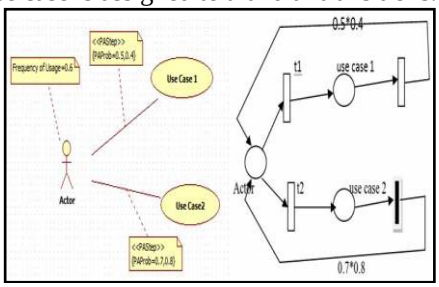


Figure1: A Use Case Diagram and a Petri Net for it.

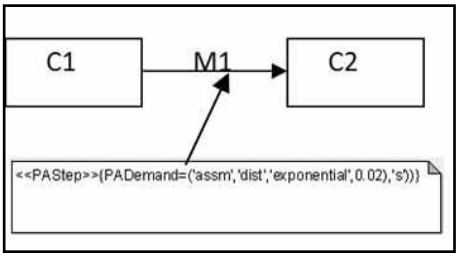


Figure2.1: Component diagram for asynchronous message

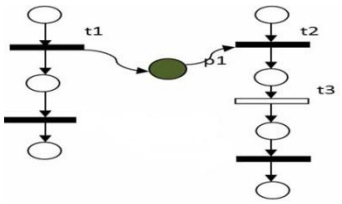


Figure2.1a: Respective Petri Net for 2.1 annoted PADemand

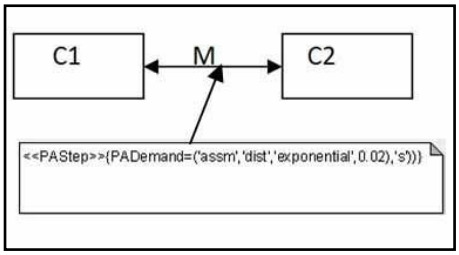


Figure 2.2 : Component diagram of Synchronous Message

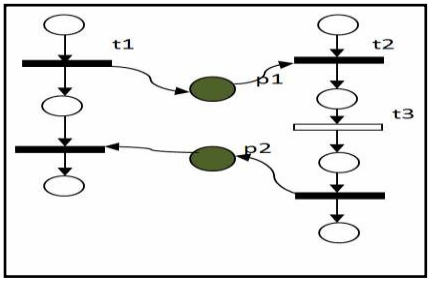


Figure2.2a:Respective Petri Net for 2.2 annoted PADemand

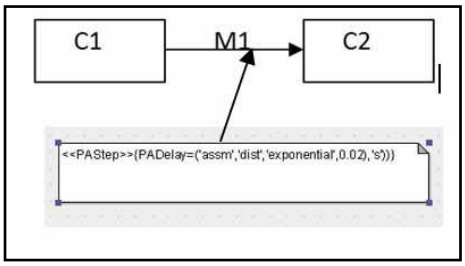


Figure 2.3 : Component Diagram annoted PADelay

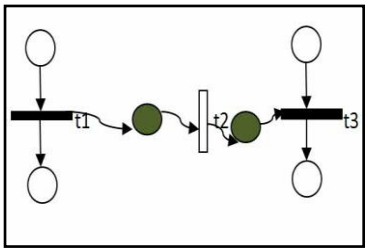
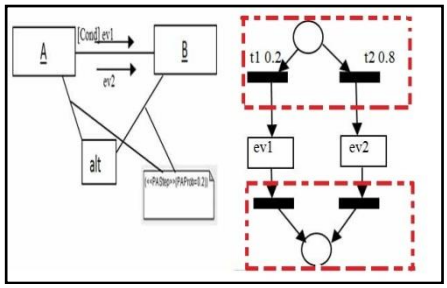


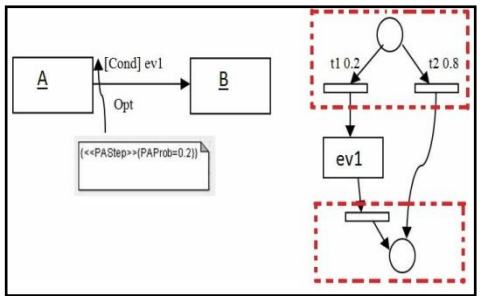
Figure2.3a: Respective Petri Net for 2.1 annoted PADelay

III. CASE STUDY

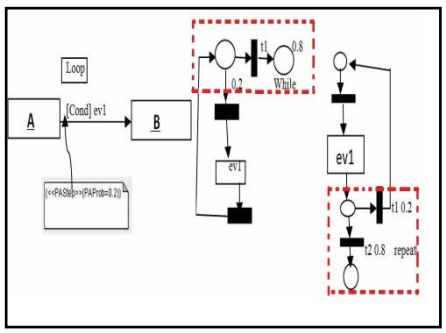
To represent the usage of our proposed algorithm, in this section we consider a single Automated Teller Machine (ATM) as an example. The use case diagram of an ATM system is shown in Figure 4.a. One of the sequence diagrams of ATM system corresponding to the use case "Identify" is shown in Figure 4.b. This diagram is used when the PIN entered by a customer is valid; the identification will be successfully done.



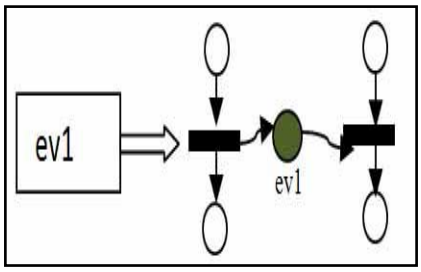
a1 a2



b1 b2



c1 c2 c3



d1

Figure 3: a1, a2, b1, b2, c1, c2, c3, d1 Petri Net Model for Component diagram structures.

According to the proposed algorithm, the equivalent Petri net of these diagrams are shown in Figure 5. ATM, bank, account and customer in Collaboration diagram which are the components in the Petri net model have been presented as separate columns. Each column presents a separate component. In this way, we distinguish the internal arcs which show transformation from one state of a component to another state, and the external arcs which show message exchange between two components. In use case diagram, customer place selects one of these use cases: balance, withdrawal or deposit. In each use case, the identification of the customer is mandatory.

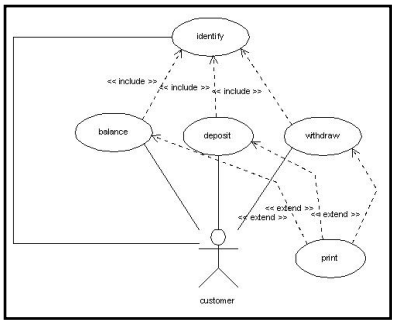


Figure 4(a) : Use Case Diagram for ATM

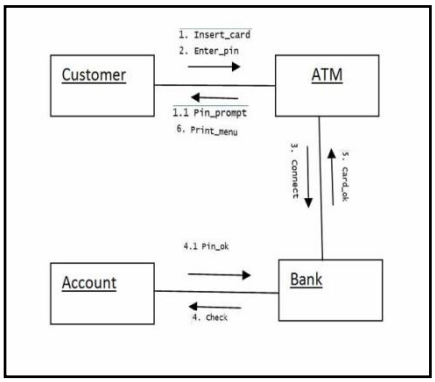


Figure 4(b) : Component Diagram of ATM

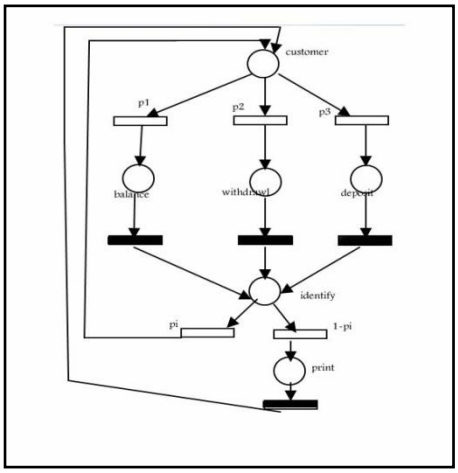


Figure5a: Petri Net of above Use case diagram(4.a)

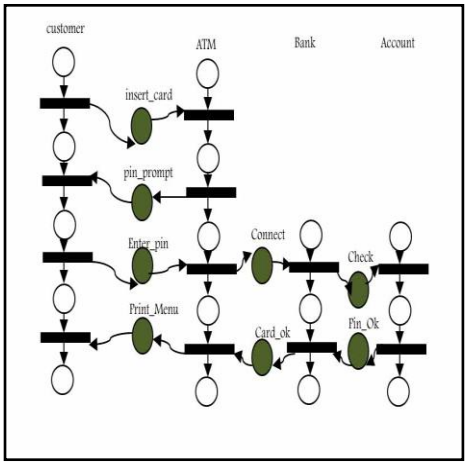


Figure 5(b): Petri Net for above component Diagram

IV. CONCLUSION

In this paper, we transformed annotated use case and collaboration diagrams with performance parameter to Petri net notations. In our further researches, we will consider the transformation of other annotated software architecture description diagrams with performance parameter to an executable model. Moreover, we can consider the annotation of additional information of other non-functional requirements to the software architecture description diagrams, as well. So, the resulting executable model can be used for evaluating those non-functional requirements.

REFERENCES

[1] Sima Emadi Engineering Department,Maybod Branch, Islamic Azad University,Yazd, Iran and Fereidoon Shams,Computer Engineering Department ,Shahid Beheshti University, Tehran- Iran Mapping annotated use case and sequence Diagrams to a Petri Net Notation for Performance Evaluation.(reference).

[2] C.U. Smith and L.G. Williams, Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software. Addison Wesley Longman Publishing Co., Inc., 2002.

[3] Bernardi S, Donatelli S, and Merseguer J. 2002, From UML Sequence Diagrams and Statecharts to Analysable Petri Net Model", In Proceedings of the 3rd International Workshop on Software and Performance, ACM, 35-45.

[4] Elkoutbi M, and Rodulf K. 1998, Modelling Interactive Systems With Hierarchical Coloured Petri Nets, In Proceeding of the Advanced Simulation Technologies Conference, Boston, MA, 432-437.

[5] Kamandi A, Abdollahi Azgomi M, Movaghar A. 2006, Transformation of UML Models into Analyzable OSAN Models, Electronic Notes in Theoretical Computer Science 159, (Elsevier, 2006), 3–22.

[6] Ourdani A, Esteban P, Paludetto M and Pascal J. C, A. 2006, Meta Modeling Approach for Sequence Diagram to Petri Nets Transformation Within the Requirements Validation Process, In 20th annual European Simulation and Modelling Conference, LAAS, Toulouse, France, ESM'2006 conference.

[7] Bernardi S , Merseguer J. 2007, Performance Evaluation of UML Design with Stochastic Well-Formed Nets, The Journal of Systems and Software 80, science direct, 1843–1865.

[8] Merseguer J, and Campos J. 2003, Exploring Roles for the UML Diagrams in Software Performance Engineering, In proceedings of the 2003 International Conference on Software Engineering Research and Practice (SERP'03), Las Vegas, Nevada, USA: CSREA Press, 43-47.

[9] Object Management Group, "UML Profile for Schedulability, Performance and Time Specification," OMG, Jan. 2005.