Capturing Performance Requirements of Real-Time Systems Using UML/MARTE Profile

Disha Khakhar and Ashalatha Nayak

Abstract Performance is a critical parameter for successful development of real-time systems since their correctness is based not only on logical behavior, but also on timeliness of output. Traditional software development methods use 'fix it later' approach where focus is on correctness of the software and performance considerations are made in the later phases of software development. If performance problems are discovered at this point, software is modified to fix performance issues. This technique does not work well for time critical systems. In order to address this problem, Model Driven Software Performance Engineering (MDSPE) approach is used to include performance analysis in the early stage of software development life cycle. Performance parameters are associated with UML model elements using UML profile for MARTE, to capture software requirement in the design phase. Annotated UML model is transformed to various performance model in order to perform analysis. By evaluating the performance model, it is possible to obtain various performance output parameters using simulation and analytical techniques. These parameters will help in evaluating alternative design for the same system. Currently, there are no approaches that investigate the issue of annotation of existing UML diagrams with MARTE profile. The proposed research focuses on capturing performance requirements by annotation of UML models using UML profile for MARTE.

Keywords UML MARTE profile · Real-time system · UML sequence diagram Software performance engineering · Performance parameters

D. Khakhar (✉) · A. Nayak
Department of Computer Science and Engineering, Manipal Institute of Technology, Manipal University, Manipal 576104, India
e-mail: disha.p.khakhar@gmail.com

A. Nayak
e-mail: asha.nayak@manipal.edu

M. Pant et al. (eds.), *Soft Computing: Theories and Applications*,
Advances in Intelligent Systems and Computing 583,
https://doi.org/10.1007/978-981-10-5687-1_62

1 Introduction

Usually, performance of software is tested later in the development phase. Unsatisfactory outcomes may lead to changes in implementation or in the worst case, and may require changes in the architecture design itself. Therefore, analyzing software performance requirement is important for evaluating various design alternatives. This can be achieved by integrating performance analysis in the early phase of software development life cycle. Unified Modeling Language (UML) is the most common way to model software requirements. OMG has standardized UML profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) in 2008 [1]. The UML profile for MARTE provides a standard for annotating UML model with performance parameters. UML/MARTE model acts as a basis for development of performance model. Performance estimation is done by evaluating software performance based on its performance model. This process involves building the performance model for performance critical scenarios followed by its evaluation using analytical and simulation techniques.

In the approach by Street et al. [2], performance modeling and analysis techniques were used for the design of object-oriented software system. It highlights the lessons learned while implementing this approach using UML profile for Schedulability, Performance and Time (SPT) along with Colored Petri Nets (CPN). An important lesson learnt was that performance analysis technique should be known prior to modeling so that only required tags are filled in the performance model.

In the approach by Traore et al. [3], Model Driven Software Performance Engineering (MDSPE) approach is presented where performance analysis is integrated with functional analysis. This paper focuses on Performance Annotation step to encapsulate performance characteristics into the software design model using UML profile for SPT.

The approach by Middleton et al. [4] describes their experience of using UML profile for MARTE to model systems with stochastic behavior using PapyrusUML editor. Demanthieu et al. [5], have highlighted how key features of MARTE profile can be used to model behavior of real-time systems. A case study related to real-time and embedded systems was developed using MARTE adopted specification. Depending on the system to be modeled (whether it is distributed), one may deal with physical or logical time. MARTE has an advantage over SPT by introducing stereotypes for time observation with explicit reference to clocks, which are useful in design of these systems.

In the approach by Akshay KC et al. [6], a case study of ATM system is taken up and UML sequence diagrams are annotated with MARTE profile to capture properties of sequential and simultaneous transactions. Data race was detected by permutation algorithm to find valid scenarios of messages retrieved from the combined fragment.

UML sequence diagram can model behavior of a system but they cannot be evaluated due to lack of formal semantics. Therefore, after the annotation of UML models with MARTE profile, UML model should be transformed to performance model for estimation of performance parameters followed by evaluation of alternative design based on performance. Currently, there are no approaches that investigate the issue of annotation of existing UML diagrams with MARTE profile. As a result, there is no systematic procedure followed for the annotation of UML diagrams, and all the existing papers directly work on transformed UML MARTE diagrams for analysis. However, during design stage, UML diagrams are adopted as de facto standard which can later be transformed for annotation purpose to capture performance requirements. Our main motivation is to capture the software performance requirements of a real-time system, in the early phase of software development life cycle, by modeling the performance requirements using UML profile for MARTE. This paper describes the use of MARTE stereotypes by taking up a case study of a time critical system.

The paper is structured as follows. Section 2 introduces the UML profile for MARTE along with description and usage of its stereotypes required to capture performance requirements. A case study of time critical systems is presented in Sect. 3 along with its UML modeling using sequence diagram. Section 4 describes the method involved in the transformation process of UML diagrams to UML/MARTE diagrams using the case study presented in Sect. 3, followed by conclusion of the research in Sect. 5.

2 Background

This section introduces the main concept of capturing performance requirement in UML model using UML profile for MARTE along with description and usage of its stereotypes.

2.1 Introduction to UML Profile for MARTE

The UML profile for MARTE adds capabilities to UML for model-driven development of Real-Time and Embedded Systems (RTES). MARTE defines the foundations for model-based description of real-time and embedded systems characterized by timing constraints, concurrency, etc. These core concepts are then refined for the purpose of modeling and analysis. Modeling part provides the support for specification of real-time and embedded characteristics to system design. Analysis part provides facilities to annotate the model with information required to perform specific analysis.

2.2 Profile Architecture of MARTE

The profile is structured around two main concerns, one to model the features of RTES and the other to annotate model so as to support analysis of system properties. These concepts provide generic description of real-time and embedded characteristics generalized under MARTE foundation. Diagrammatic representation of basic MARTE architecture is shown in Fig. 1.

Each package consists of various sub-packages which addresses specific concerns like timing, resource allocation etc.

MARTE Foundation Model—It is a shared package which addresses common concerns while describing use of concurrent resources and time. It consists of following sub-packages:

- **Non-Functional Properties Modeling**—Application properties are grouped into two categories: functional properties, which are concerned with what the system does at run-time; and Non-Functional Properties (NFPs), which describe how well the system performs its functions. NFPs provide information about various properties such as throughput, overhead, delays, memory usage, etc. This package provides mechanism for specification of NFPs in UML/MARTE model.
- **Time Modeling**—Real-time systems are associated with timing constraints. This package provides framework for representing time and time-related concepts of real-time system in UML/MARTE model.

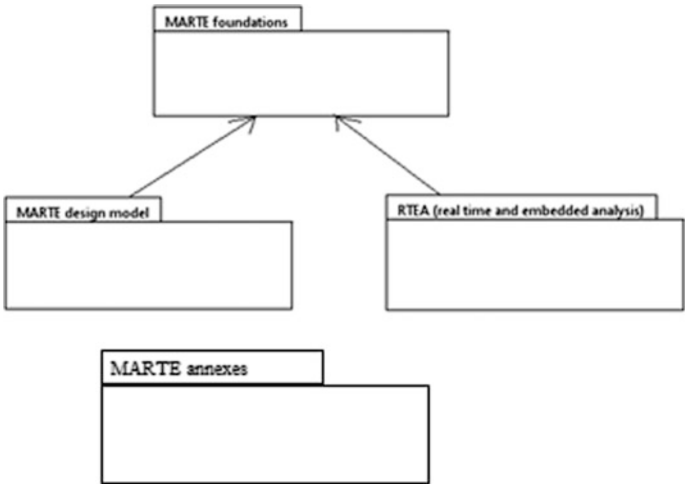

Fig. 1 Basic architecture of MARTE profile

- **General Resource Modeling**—It specifies how to describe resource model at system level. It includes features which are required for dealing with modeling of both software (operating system, etc.) and hardware (memory unit, communication channel, etc.).

MARTE Design Model—It defines the MARTE concepts for model-based design of RTES. It consists of following sub-packages:

- **High-Level Application Modeling**—It provides high-level modeling concepts to deal with real-time and embedded features modeling.
- **Software Resource Modeling (SRM)**—There are two approaches to the design of RTES applications: sequential based design approach and multitask based design approach. Applications designed with multitasking approach have specific execution mechanism on platforms requiring specific execution support. This support provides a set of resources and services for real-time features of an application. It is possible to describe the structure of such support by using modeling artifacts specified by SRM.

Real-Time and Embedded Analysis—It is focused on model-based analysis. It does not define new analysis technologies, but additional information for annotation of models for analysis. It consists of following sub-packages:

- **Generic Quantitative Analysis Modeling**—The generic analysis domain includes specialized domains in which the analysis is based on the software behavior, such as performance, availability, etc. Quantitative analysis (i.e. analysis of non-functional properties (NFPs)) techniques determine the values of ‘output NFPs’ based on data provided as ‘input NFPs’.
- **Performance Analysis Modeling**—It describes the analysis of temporal properties of soft real-time systems, including web-based services, multimedia, networked services, etc. for which performance measures are statistical such as mean throughput or delay.

MARTE Annexes—Annexes contain useful information about various value specification languages provided by MARTE profile. Value specification language deals with specification of parameters, expressions, relationship between different variables in textual form.

2.3 MARTE Stereotypes

The UML profile for MARTE defines a set of stereotypes which allows us to map model elements to characteristics of real-time system. Stereotypes are associated with attributes that gives values for properties which are needed in order to carry out the analysis. Stereotypes used for the proposed research are summarized in Table 1.

Table 1 MARTE stereotypes

Stereotype	Description	Attributes
≪PaStep≫	A step is a unit of a scenario. It is a basic sequential execution step on a host processor	execTime—time for execution (response time minus any initial scheduling delays)
≪PaCommStep≫	A CommStep is an operation which conveys a message from one locale to another (e.g.: response from server to client). The message conveyance may be executed by a combination of host middleware and network services	msgSize—the size of the message to be transmitted by the step
≪RtService≫	It can specify the real-time features described by its attributes	concPolicy—concurrency policy used for the real-time service (reader/writer) isAtomic—when true, implies that the RtService executes as one indivisible unit, non-interleaved with other RtServices
≪Acquire≫	It is used to acquire a protected resource	isBlocking—if true it indicates that any attempt to acquire the resource may result in a blocking situation if it is not available. If false it indicates the unavailability of the protected resource will not block the caller but it will be returned as part of the service results instead
≪Release≫	It is used to free an acquired protected resource	nil
≪GaWorkload Event≫	It is a stream of events that initiate system level behavior	pattern—this attribute defines a pattern of arrival events. It can be periodic, aperiodic, irregular, etc.
≪SWConcurrent Resource≫	This resource defines entities, which may execute concurrent instructions while providing an executing context to a routine	nil
≪PaLogical Resource≫	A PaLogicalResource is a resource that can be acquired and released explicitly by AcqStep or RelStep. It may be a single unit resource, as a mutex or exclusive lock, or have multiple units, as a buffer pool or an access token pool	poolSize—the number of units of the resource
≪SharedDataCom Resource≫	It defines specific resource used to share the same area of memory among concurrent resources. They allow concurrent resources to exchange safely information by reading and writing the same area in memory	nil

3 Case Study and Its UML Modeling

To illustrate the modeling elements introduced above, a case study named "CPU Allotment" is presented in this section. First, the requirements are stated through problem description followed by modeling of the system. System modeling includes:

- Identifying use-case to depict system functionality
- Sequence diagram to understand system behavior for each use-case

3.1 Description of CPU Allotment Case Study

CPU Allotment is a web-based application that allows students to reserve CPU in advance every time they want to practice in lab. They can register to obtain login credentials required for reserving CPU, and each student can reserve only one CPU per slot. This web-based application allows a student to select date and slot corresponding to which available CPUs are displayed. The required CPU can be reserved by selecting and clicking on confirm button.

3.2 Modeling the System Using Use-Case Diagram

The above requirements can be captured by a use-case diagram shown in Fig. 2, which depicts system functionalities by means of use-cases.

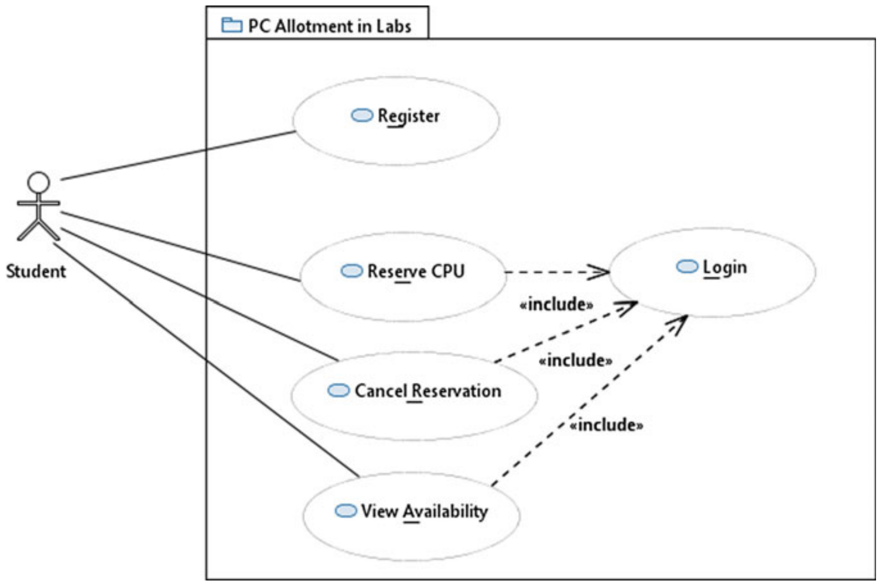

Fig. 2 Use-case diagram for CPU allotment

The use-cases are described in brief below:

- Register—Allows a first time user to obtain login credentials
- Login—Verifies whether a student is allowed to use system functionalities
- Reserve CPU—Allows a student to reserve CPU for selected slot
- Cancel reservation—Allows a user to cancel reservation for a particular slot
- View Availability—System displays available CPUs for selected slot

From the use-cases, Register and Reserve CPU are identified as performance critical. For Register use-case, whenever a student tries to set his/her UserID the system should verify that it is unique in real time. When multiple users are trying to reserve CPU concurrently, the system should allot CPU to only one user. Hence, these two use-cases involve concurrency and performance criteria for the system to respond in real time for correct functioning.

3.3 Modeling the System Using Sequence Diagram

It is assumed that application runs concurrently serving the request of users by creating a new thread for each incoming request. A database which maintains reservation/user information is shared by concurrently executing threads. Multiple read operations are allowed while only one thread can write to the database at a time. A thread cannot write while some other thread is reading from the database. Therefore, before writing, each thread has to acquire the lock and release it after writing. Sequence diagram depicting system behavior is shown Figs. 3 and 4.

A. Register

Register use-case allows a user to set Login credentials for the website. The system should verify, in real time that the userID provided is unique. If a user with same userID already exists, then it should prompt the user to change it. Once uniqueness is verified, entry corresponding to the new user should be added to the database. The sequence of messages involved for executing Register use-case scenario is shown as Sequence Diagram in Fig. 3.

B. Reserve CPU

Reserve CPU use-case allows a user to select a CPU to be reserved for a specific date and slot. When multiple users are trying to reserve the same CPU, the system should make sure that only one of them is successful. This is done by restricting write operation by multiple threads to database. Only one thread is allowed to write at a time while other threads will wait in a queue for gaining access to database. The sequence of messages involved for executing Reserve CPU use-case scenario is shown as sequence diagram in Fig. 4.

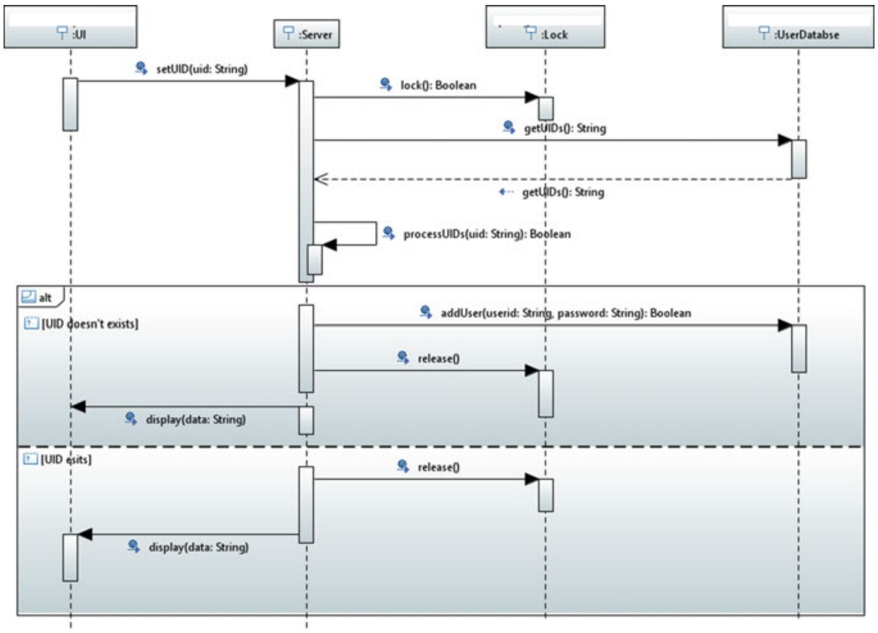

Fig. 3 Sequence diagram for Register

4 Methodology

Using UML diagrams as input, performance requirement of the system is captured by annotating UML model with MARTE profile. The process of adding stereotype labels to UML model for expressing performance (or similar quantitative) concepts is called as annotation. This process involves following steps:

1. Identifying performance critical scenarios of the system.
2. Selecting the stereotypes required to map UML model elements to characteristics of the system.
3. Defining values for stereotype attributes (tagged values) which represents quantitative properties of the system.
4. Associating stereotype labels, along with tagged values to elements of UML model.

The output of this step is annotated UML/MARTE model depicting system behavior and its real-time characteristics. Since UML lacks formal semantics, it is not possible to apply mathematical techniques directly to evaluate performance. Therefore, a transformation to performance model is required for analysis.

This section explains how each step of methodology is implemented for the case study described in Sect. 3. We begin with describing how stereotypes are applied to UML models to obtain annotated UML/MARTE Sequence Diagram.

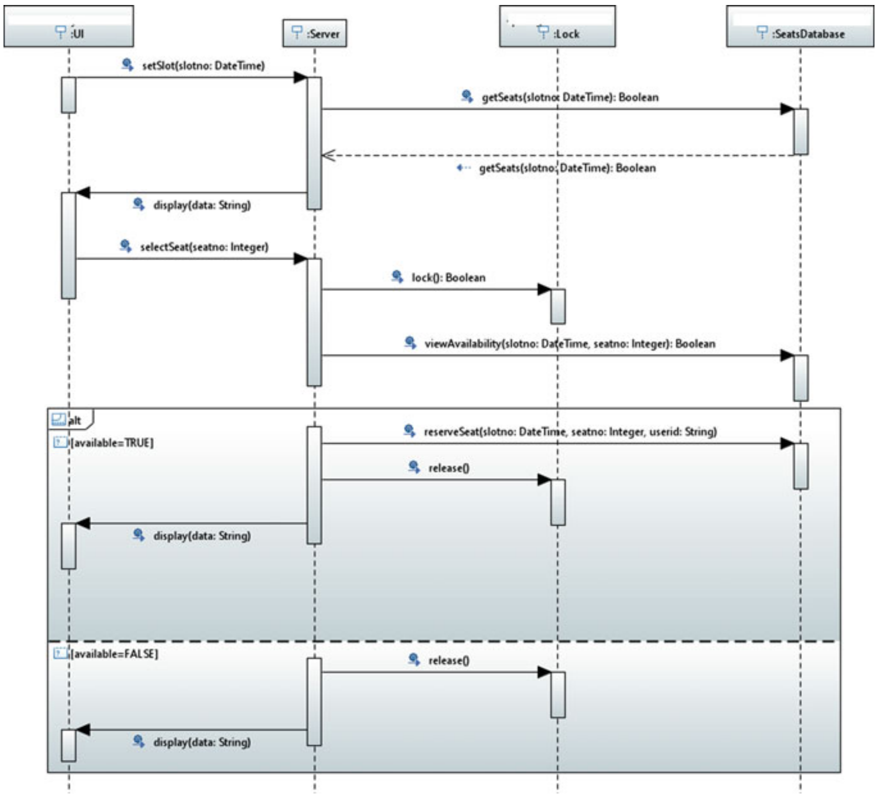
:UI
:Server
:Lock
:SeatsDatabase
setSlot(slotno: DateTime)
getSeats(slotno: DateTime): Boolean
getSeats(slotno: DateTime): Boolean
display(data: String)
selectSeat(seatno: Integer)
lock(): Boolean
viewAvailability(slotno: DateTime, seatno: Integer): Boolean
alt
[available=TRUE]
reserveSeat(slotno: DateTime, seatno: Integer, userid: String)
release()
display(data: String)
[available=FALSE]
release()
display(data: String)

Fig. 4 Sequence diagram for Reserve CPU

Referring to Table 1, the User interface is labeled as «SwConcurrent Resource» since concurrent users interact with the application through it. Lock is annotated with «PaLogicalResource» with pool size as 1 because only one thread can acquire it at a time. Processing step annotated with «PaStep» indicates that this step is performance critical with ‘execTime’ attribute denoting time required for execution of this step. Function call to services which require response in real time are annotated with «RtService» stereotype. «RtService» is associated with the type of operation (read or write) being performed by using ‘concPolicy’ attribute. If the required operation is writing, then it is to be performed atomically specified by ‘isAtomic’ attribute. Response from server to client is annotated with «PaComm Step» because it is sent via a communication channel between them. Database is associated with «sharedDataCommResource» stereotype since it is shared among multiple concurrently executing threads. Annotated sequence diagram with MARTE profile for use-cases Reserve CPU and Register is shown in Figs. 5 and 6.

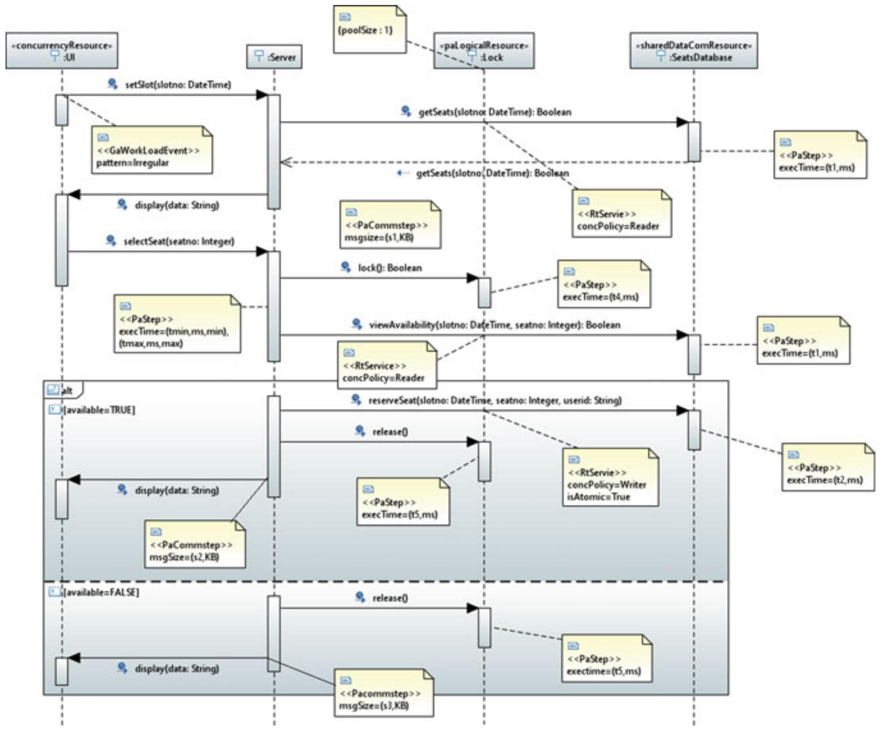
«concurrencyResource» :UI
:Server
{poolSize : 1}
«paLogicalResource» :Lock
«sharedDataComResource» :SeatsDatabase
setSlot(slotno: DateTime)
getSeats(slotno: DateTime): Boolean
<<GaWorkLoadEvent>> pattern=Irregular
<<PaStep>> execTime=(t1,ms)
getSeats(slotno: DateTime): Boolean
display(data: String)
<<PaCommstep>> msgsize=(s1,KB)
<<RtServie>> concPolicy=Reader
selectSeat(seatno: Integer)
lock(): Boolean
<<PaStep>> execTime=(t4,ms)
<<PaStep>> execTime=(tmin,ms,min), (tmax,ms,max)
viewAvailability(slotno: DateTime, seatno: Integer): Boolean
<<RtService>> concPolicy=Reader
<<PaStep>> execTime=(t1,ms)
alt
[available=TRUE]
reserveSeat(slotno: DateTime, seatno: Integer, userid: String)
release()
<<RtService>> concPolicy=Writer isAtomic=True
<<PaStep>> execTime=(t2,ms)
display(data: String)
<<PaStep>> execTime=(t5,ms)
<<PaCommstep>> msgSize=(s2,KB)
[available=FALSE]
release()
<<PaStep>> exectime=(t5,ms)
display(data: String)
<<Pacommstep>> msgSize=(s3,KB)

Fig. 5 Annotated sequence diagram for Reserve CPU use-case

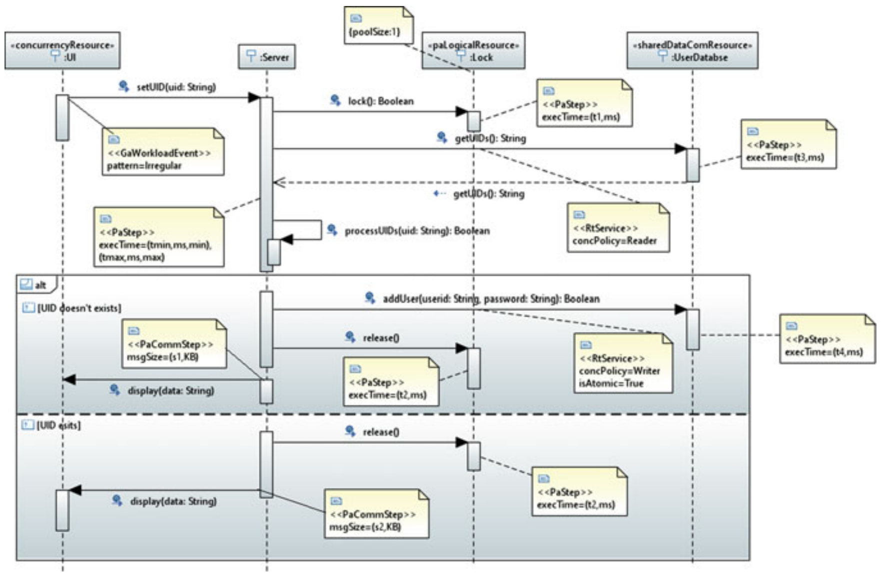
«concurrencyResource» :UI
:Server
{poolSize:1}
«paLogicalResource» :Lock
«sharedDataComResource» :UserDatabse
setUID(uid: String)
lock(): Boolean
<<PaStep>> execTime=(t1,ms)
getUIDs(): String
<<PaStep>> execTime=(t3,ms)
<<GaWorkloadEvent>> pattern=Irregular
getUIDs(): String
<<PaStep>> execTime=(tmin,ms,min), (tmax,ms,max)
processUIDs(uid: String): Boolean
<<RtService>> concPolicy=Reader
alt
[UID doesn't exists]
addUser(userid: String, password: String): Boolean
<<PaCommStep>> msgSize=(s1,KB)
release()
<<RtService>> concPolicy=Writer isAtomic=True
<<PaStep>> execTime=(t4,ms)
display(data: String)
<<PaStep>> execTime=(t2,ms)
[UID exits]
release()
display(data: String)
<<PaStep>> execTime=(t2,ms)
<<PaCommStep>> msgSize=(s2,KB)

Fig. 6 Annotated sequence diagram for Register use-case

5 Conclusion

In order to demonstrate the transformation of UML diagrams to capture performance requirements, a case study of time critical system has been stated. Performance critical scenarios identified in the given case study are modeled using UML. Use-case diagrams depicting system functionalities and sequence diagrams depicting system behavior are designed. Capturing performance requirement in the design phase has increasingly become critical for real-time applications. MARTE profile allows construction of models that may be used to make quantitative predictions regarding real-time and embedded characteristics of systems, by allowing annotation of UML diagrams, to capture performance requirements. Therefore, it is important to understand the procedure involved in annotation of UML diagrams. This paper addresses the concern of annotating UML diagrams with MARTE profile by providing detailed explanation involving the type and usage of MARTE stereotypes. Since UML models cannot be analyzed mathematically, future work will include transforming annotated sequence diagrams to performance models for performance estimation and evaluation.

References

1. OMG.: The UML profile for Modeling and Analysis of Real Time and Embedded System (MARTE). Available: http://www.omgmarte.org/
2. Street, J.A., Pettit, R.G.: Lessons learned applying performance modeling and analysis techniques. In: Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'06), pp. 7 (2006)
3. Traore, I., Woungang, I., Ahmed, A., Obaidat, M.S.: Software Performance Modeling using the UML: a Case Study, J. Networks, **7**(1), 4–20 (2012)
4. Middleton, S.E., Servin, A., Zlatev, Z., Nasser, B., Papay, J., Boniface, M.: Experiences using the UML profile for MARTE to stochastically model post-production interactive applications. In: eChallenges e-2010 Conference, pp. 1–8 (2010)
5. Demathieu, S., Thomas, F., AndrÃ c, C., GÃ crard, S., Terrier, F.: First experiments using the UML profile for MARTE. In: 1th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), pp. 50–57 (2008)
6. Akshay, K.C., Nayak, A., Muniyal, B.: Modeling data races using UML/MARTE profile. In: International Conference on Advances in Computing, Communications and Informatics (ICACCI 2014), pp. 238–244 (2014)