Translation from UML to Markov Model: A Performance modeling Framework for Managing Behavior of Multiple Collaborative Sessions and Instances

Razib Hayat Khan  
Norwegian University of Science & Technology (NTNU)  
7491, Trondheim, Norway  
rkhan@item.ntnu.no

Poul E. Heegaard  
Norwegian University of Science & Technology (NTNU)  
7491, Trondheim, Norway  
oul.heegaard@item.ntnu.no

Abstract—Performance evaluation of distributed system is always an intricate undertaking where system behavior is normally distributed among several components those are physically distributed. Bearing this concept in mind, we delineate a performance modeling framework for a distributed system that proposes a translation process from high level UML notation to Markov model and solves the model for relevant performance metrics. To capture the system dynamics through our proposed framework we outline a specification style that focuses on UML collaboration and activity as reusable specification building blocks. To present the UML specification style we focus on how to coordinate explicitly multiple collaborative sessions occurring at the same time. Design alternatives of system architecture are considered to generate the performance model to show how these design alternatives thus affect the system performance under different work load. The proposed performance modeling framework provides prediction result of a system such as mean response time and resource utilization. The applicability of our proposed framework is demonstrated in the context of performance modeling of a distributed system.

Keywords—UML, Collaborative Session, Markov model

I. INTRODUCTION

Distributed system is one of the main streams of information and communication technology arena. Modeling, developing, and implementation of such complex systems are always a difficult endeavor. Likewise performance evaluation is also a great concern of such complex system to evaluate whether the system meets the performance related system requirements. However in a distributed system, system behavior is normally distributed among several objects. The overall behavior of the system is composed of the partial behavior of the objects of the system. So it is obvious to capture the behavior of the distributed objects of the system to evaluate the performance of the overall system. We therefore adopt UML collaboration and activity oriented approach as UML is the most widely used modeling language which models both the system requirements and qualitative behavior through different notations. Collaboration and activity diagrams are utilized to demonstrate the overall system behavior by defining both the structure of the partial object behavior as well as the interaction between them as reusable specification building blocks and later this UML specification style is applied to generate the Markov model by our proposed performance modeling framework. UML collaboration and activity provides a tremendous modeling framework containing several interesting properties [1]. Firstly collaborations and activity model the concept of service provided by the system very nicely. They define structure of partial object behaviors, the collaboration roles and enable a precise definition of the overall system behavior. They also delineate the way to compose services by means of collaboration uses and role bindings [2].

In addition, the proposed modeling framework considers design alternatives of system architecture to generate performance model of the system to show the performance affect because of the changing of system architecture and to help to find out the better system architecture candidate to fulfill certain performance goal at the early stage of the system development process. Abstract view of the system architecture is captured by the UML deployment diagram which defines the execution architecture of the systems by identifying the system components and the assignment of software artifacts to those identified system components [1]. Considering the system architecture while generating the performance model also resolves the bottleneck of system performance by finding a better allocation of service components to the physical component of the system.

The Unified Modeling Language (UML) is a widely accepted modeling language to model the system behavior [1]. But it is indispensable to extend the UML model to incorporate the performance related quality of service (QoS) information to allow modeling and evaluating the properties of a system like throughput, utilization, mean response time. So the UML models are annotated according to the Profile for Schedulability, Performance, and Time (SPT) [3] to include quantitative system parameters in the model.

Markov models, queueing networks, stochastic process algebras and stochastic petri net are probably the best studied performance modeling techniques [4]. Among all of them, we will choose Markov model as the performance model generated by our proposed framework for providing performance prediction result of a system due to its modeling generality, its well-developed numerical modeling analysis techniques, its ability to preserve the original architecture of the system, to facilitate any modification according to the feedback from performance evaluation and the existence of analysis tools.

Numbers of efforts have been made already to generate a performance model from the system design specification. Kähkipuro developed a performance modeling framework to generate and solve queue network with simultaneous resource possessions from the high level UML notations [5]. Lopez-Grao et al. proposed a conversion method from
annotated UML activity diagram to stochastic petrinet model [6]. Abdullatif and Pooly presented a method for providing computer support for extracting Markov chains from a performance annotated UML sequence diagram [7]. The framework presented here is the first known approach that introduces a new specification style utilizing UML behavioral diagrams as reusable specification building block for managing multiple collaborative sessions that executed at the same time which is later used for generating performance model to produce performance prediction result at early stage of the system development process.

The objective of the paper is to provide an extensive performance modeling framework that provides a translation process to generate markov performance model from system design specification captured by the UML behavioral diagram for multiple collaborative sessions that executed at the same time and for design alternatives of the system architecture and later solves the model for relevant performance metrics to demonstrate performance prediction results at early stage of the system development life cycle. The paper is organized as follows: section II introduces our proposed performance modeling framework, application example is demonstrated in section III and section IV delineates the conclusion with future works.

II. PROPOSED PERFORMANCE MODELING FRAMEWORK

Our proposed performance modeling framework utilizes the tool suite Arctis which is integrated as plug-ins into the eclipse IDE [8]. The proposed framework is composed of 7 steps shown in figure 1 where steps 1 and 2 are the parts of Arctis tool suite. Arctis focuses on the abstract, reusable service specifications that are composed form UML 2.2 collaborations and activities. It uses collaborative building blocks as reusable specification units to create comprehensive services through composition. To support the construction of building block consisting of collaborations and activities, Arctis offers special actions and wizards. In addition a number of inspections ensure the syntactic consistency of building blocks. A developer first consults a library to check if an already existing collaboration block or a collaboration of several blocks solves a certain task. Missing blocks can also be created from scratch and stored in the library for later reuse. The building blocks are expressed as UML models. The structural aspect, for example the service component and their multiplicities, is expressed by means of UML 2.2 collaborations. For the detailed internal behavior, UML 2.2 activities have been used. They express the local behavior of each of the service components as well as their necessary interactions in a compact and self-contained way using explicit control flows [8]. Moreover the building blocks are combined into more comprehensive service by composition. For this composition, Arctis use UML 2.2 collaborations and activities as well. While collaborations provide a good overview of the structural aspect of the composition, i.e., which sub-services are reused and how their collaboration roles are bound, activities express the detailed coupling of their respective behaviors. To reason about the correctness of the specifications, we introduce formal reasoning on the level of collaborative service specifications using temporal logic specification style cTLA/c(c for collaborative) [8].

The step of our proposed modeling framework is described as follows:

Step1 - Construction of collaborative building block

The proposed framework utilizes collaboration as main specification units. The specifications for collaborations are given as coherent, self-contained reusable building blocks. The structure of the building block is described by UML 2.2 collaboration. If the building block is elementary it only declares the participants (as collaboration roles) and connection between them. If it is composite, it may additionally refer to other collaborations between the collaboration roles by means of collaboration uses. The internal behavior of building block is described by UML activity. It is declared as the classifier behavior of the collaboration and has one activity partition for each collaboration role in the structural description. For each collaboration use, the activity declares a corresponding call behavior action referring to the activities of the employed building blocks.

Depending on the number of participants, connectivity to other blocks and level of decomposition, we distinguish three different kinds of building blocks [8]:

- The most general building block is collaboration with two or more participants providing functionality that is intended to be composed with other functionality. We refer to such a building block as service collaboration.
- Building blocks that involve only local behavior of one participant are referred to as activity blocks. They are represented by activities.
- A special building block is system collaboration, which is collaboration on the highest composition level. In contrast to a service, a system is closed and cannot be composed with other building blocks.

Step 2 – Composition of building block using UML collaboration & activity

To generate the performance model, the structural information about how the collaborations are composed is not sufficient.
It is necessary to specify the detailed behavior of how the different events of collaborations are composed so that the desired overall system behavior can be obtained. For the composition, UML collaborations and activities are used complementary to each other; UML collaborations focus on the role binding and structural aspect, while UML activities complement this by covering also the behavioral aspect for composition. For this purpose, call behavior actions are used. Each sub-service is represented by a call behavior action referring to respective activity of the building blocks. Each call behavior action represents an instance of a building block. For each activity parameter nodes of the referred activity, a call behavior action declares a corresponding pin. Pins have the same symbol as activity parameter nodes to represent them on the frame of a call behavior action. Arbitrary logic between pins may be used to synchronize the events of the building block and transfer data between them. By connecting the individual input and output pins of the call behavior actions, the events occurring in different collaborations can be coupled with each other.

There are different kinds of pins described as follows [8]:
- Starting pins activate the building block, which is the precondition of any internal behavior.
- Streaming pin may pass tokens throughout the active phase of the building block.
- Terminating pins mark the end of the block’s behavior.

If collaborations is started and terminated via several alternative pins, they must belong to different parameter sets. This is visualized in UML by an additional box around the corresponding node.

To present the UML specification style we focus on how to coordinate explicitly multiple collaborative sessions occurring at the same time. To reflect the multiplicity of the service components, their partitions are represented by several layers. This multiplicity of partitions implies a certain multiplicity of collaborations that have to be coordinated by the service component. A token arriving from any of the collaboration instances simply enters the partition. Vice-versa, when a token should enter a specific collaboration instance from the partition, we need to determine which instance should receive the token. UML does not give any means to select such session. Therefore we include one selection operator in the execution profile. To represent the overall system behavior for multiple session instances, the different sessions must be distinguished at run time.

**Step 3 - UML deployment diagram & stating relation between system component & collaboration**

Deployment diagram can be used to define the execution architecture of systems by identifying the system components and the assignment of software artifacts to those identified system components [1]. After designing the deployment diagram the relations between system component and collaboration will be delineated to describe the service delivered by the system. The service is delivered by the joint behavior of the system components which may be physically distributed. The partial behavior of the component utilized to realize the collaboration is represented by the collaboration role. In this way it is possible to expose direct mapping between the collaboration roles to the system components to show the probable deployment of service components in the physical nodes of the system.

**Step 4 – Annotating the UML model**

Performance information is incorporated into the UML activity diagram and deployment diagram according to UML Profile for Schedulability, Performance & Time [3]. UML model is needed to annotate for evaluating system performance by performance model solver for relevant performance metrics through our proposed framework.

**Step 5 – State marking and reachability graph**

To generate the reachability graph, we consider the behavior of the system components which are the subject of the bottleneck of the system performance and the states of the system will be generated based on the activity performed by those components which are shown by their internal behavior explained in step 1. This section involves a state marking procedure that will mark the state of the system for each step executed and produce a reachability graph of the whole system. Each of the marked state will represent a state of the overall Markov chain. The method for state marking of the system is as follows: first an initial situation is marked, which defines as the initial state of the system before executing the first step. Then when a step will be executed, the activity performed by the system component in this step...
will be marked as performance model state from where a new state may be generated or return back to an already marked state with a transition rate. This procedure will be continued until all the steps are executed. After that the reachability graph of the system will be produced. The execution of steps will be outlined in the composition of building blocks while describing the overall system behavior. The number and sequence of execution of steps will differ according to the system design specification.

**Step 6 – Generating markov model**

If each of the reachable marking is represented as a node and each edge between the nodes level with the trigger of their change by the transition rate (mentioned in annotated UML model), we can generate the markov model of the system.

**Step 7 – Evaluate model**

Generated markov model will be used as input for the SHARPE tool [9] to generate performance prediction result of the system.

## III. Application Example

As a representative example, we introduce a scenario description to show the applicability of our proposed framework in designing, modeling and performance evaluating of a distributed system. Several users are equipped with cell phones or PDAs want to receive weather information of their current location using their hand held devices. The user request is first transferred to location servers through base transceiver station to retrieve the location information of the user. The location information is then transferred to weather servers for retrieving the weather information according to the location of the user.

Figure 2 defines this scenario as UML 2.2 collaboration. Participants in the system are users, mobile terminals, base transceiver stations, location servers, weather servers which are represented by the collaboration roles user, MT, BTS, LS, and WS. The users are the part of the environment and therefore labeled as <<external>>. The default multiplicity of the users, mobile terminals, base transceiver stations, location servers, weather servers are many which are denoted by [1..*]. The interactions between the collaboration roles are represented by the collaboration use such as mobile terminal and BTS interact through t: transfer, BTS and location server, weather server interact through collaboration uses l: request location info, w: request weather info, while the user interacts with the mobile terminal by collaboration use g: generate request.

### Step 1 - Construction of collaborative building block:

This step defines the formulation of the building blocks in form of collaboration as major specification unit of our framework shown in Fig.2. The structure of the building block is defined by the UML collaboration shown in Fig.3 (a). The building block declares the participants as collaboration role and connection between them. While the UML collaboration describes the structural aspect of the composed service, the internal behavior of the collaboration is described by the UML activity [2]. Hereby collaborations of figure 2 are modeled by a call behavior action referring to the activity to describe the behavior of the corresponding collaboration [10]. Activity diagram presents complete behavior in a quite compact form and may define connections to other behaviors via input and output pins. Here we specify the behavior of one user request to show how the request is generated from user’s mobile terminal and processed by the BTS, location server and weather server and later compose this behavior for multiple user requests to show how the requests will be processed by the multiple BTS, multiple instance of location servers and weather servers so that the overall system behavior can be delineated. To deliver the requested information to the user through his/her mobile terminal, BTS participates in the collaboration Request Location Info together with the location server and Request Weather info together with weather server. These are specified by the collaboration l: Request Location Info and w: Request Weather info where the BTS plays the role client and the location server and weather server play the role server. The behavior of the collaboration is described by the UML activity in figure 3 (b) where activity is divided into two partitions: one for each collaboration role (Client & Server). The activity is started on the client side when the user request is provided as parameter u_req at the input pin. The u_req directly sent to the location server where it is converted into a database request by the call behavior action processing. After that it is the task of the collaboration between the server and the database to provide the stored information. To get the information the request leaves the activity Request Location info and the server waits for the reception of response. This is modeled with the input and output pins request and response.

![Figure 2. Collaboration Diagram](image-url)
After getting the response the result $l_{\text{info}}$ (location information) is delivered to the corresponding output pin in the client side by the call behavior action delivery and the activity is terminated. Here we describe the behavior of collaboration Request Location info. Likewise we can describe the behavior of Request Weather info through activity partition of client and server where location information of user is forwarded by the client to request server to retrieve weather information of the user location.

**Step 2 – Composition of building block:** Figure 4 shows the activity diagram for our system to highlight the overall behavior of the system by composing all the building blocks via several pins. The initial node (●) marks the starting of the activities. The activity is started on the client side. When a user service request is generated via mobile terminal, $g$: Generate request will transfer the user service request as parameter $u_{\text{req}}$ to the BTS via collaboration $t$: Transfer. Once arrived at the BTS request for location information is forwarded to the location server represented by activity Request location info. Location server (LS) makes a database request which is modeled by $d1$: DBRetrieve and terminates with result $l_{\text{info}}$ (Location information). After getting the location information, request for weather information according to user current location is forwarded by the BTS to the weather server (WS) represented by activity Request weather info. Weather server makes a database request which is modeled by $d2$: DBRetrieve and terminates with result $w_{\text{info}}$ (Weather information). After that, the final result is transferred to the user hand held device by BTS via collaboration $t$: Transfer.

The structure of collaborations as well as the way to couple them facilitates the reuse of activities. For example both the collaboration $d1$ and $d2$ are identical and can be instantiated from single collaboration type. Moreover the collaboration $l$ and $w$ have very similar behavior and can be based on the same UML template. Thus, systems of a specific domain can often be composed of reoccurring building blocks by reusing them [10].

From the viewpoint of one user, one location server and one weather server, there is exactly one collaboration session for the collaboration use $t$, $l$ and $w$ towards the BTS at certain time. This can be handled easily with the UML activity diagram in their standard form. But one BTS has to maintain several sessions with each of the user and each of the location and weather server at certain time. From the viewpoint of one BTS, several instances of the collaboration use $t$, $l$ and $w$ are executed at the same time; one instance for each user, location server and weather server. From the viewpoint of BTS, the collaborations that it participates are called multi-session collaboration. We express this by applying a stereotype <<multi-session>> to the call behavior actions and represents it graphically by multiple borders in those partitions where sessions are multiple shown in Figure 4 [2]. One of the important issues here is that how the different instances of collaborations may be distinguished and coordinated, so that desired overall system behavior is obtained. Therefore we need a selection mechanism so that selection of sessions must take place whenever a token enters a multi-session sub-collaboration and the overall system behavior can be reflected correctly for multiple instance of session for users, location servers and weather servers. While in some cases we may want to address all of the sessions, in other ones we like to select only a subset or one particular session. The UML standard however does not elaborate this matter. This is too restrictive, as most system exhibit patterns with several executions going on at a time that possibly need coordination [2]. We therefore added the new operators select to our execution profile. To represent the overall system behavior for multiple session instances, the different sessions must be distinguished at run time. This resembles the well-known session pattern [11] which is found in client/server communication, where server has some kind of identifier to distinguish different sessions. For our case, we can assign an ID to the each request (req_id) to identify the session instance of the transfer, request location info and request weather info collaboration. When BTS receives the response form the location server (LS) about the location of the user, a token leaves output pin $l_{\text{info}}$ and enters $w$: request weather info. Here we have to select the session instance of the user so that user request can be successfully processed. Likewise selection of session instance of user should be chosen to deliver the result to the user hand held devices. As they are distinguished by the request id we leave this number as attribute req_id inside the token and extract it by writing select one : id = req_id. The complete EBNF definition for session selection is given in Figure 5.

**Step 3 - UML deployment diagram & stating relation between system component & collaboration:** We consider two design alternatives of system architecture to
demonstrate the relationship between collaboration and system component.

In the first case the identified system components by our deployment diagram shown in Figure 6a are Mobile terminal, Base transceiver station, Location server and Weather server. The artifacts locationserverprocess.exe and weatherserverprocess.exe are assigned successively to location server and weather server. After designing the deployment diagram the relationship between system component and collaboration will be delineated to describe the service delivered by the system. The service is delivered by the joint behavior of the system components which may be physically distributed. The partial behavior of the component utilized to realize the collaboration is represented by the collaboration role. For our defined system description the behavior of the components Mobile terminal, Base transceiver station, Location server, Weather server are represented by the collaboration roles MT, BTS, LS & WS to utilize collaboration t: transfer and the behavior of the component application server is represented jointly by the collaboration role LS and WS to utilize collaboration l: request location info and w: request weather info shown in Figure 8. In the second case the mapping between system component & collaboration role is generalized into one to many relations.

Step 4 – Annotating the UML model: Performance information is incorporated into the UML activity diagram in Figure 4 and deployment diagram in Figure 6 according to the UML Profile for Schedulability, Performance and Time [4] to enable system performance to be evaluated by performance model solver for relevant performance metrics through our proposed framework. We use the stereotype PAcontext, PAopenLoad, PAhost, PAstep and the tag value PAoccurrence, PASchdPolicy, PArespTime and PAinterval [3].
A performance context can be modeled by an activity graph that is stereotyped as a PAcontext. This means that all interactions specified in that activity graph represent scenarios in the sense of this profile (shown in Fig.4). A PAopenLoad is modeled as a stream of requests that arrive at a given rate in predetermined pattern with PAoccurrence. A PAhost models a processing resource with tags PAschdPolicy defining the policy by which access to the resource is controlled. A PAstep models a scenario step with tags PArespTime defining a step’s response time and PAinterval defines time interval between successive repetitions of a step.

**Step 5 – State marking and reachability graph:** To generate the performance model by our proposed framework we will consider the above two design alternatives of the system architecture where the components have multiple instances. For our example scenario and first variation of deployment diagram, we will consider the participants location server and weather server as limiting factor for the system performance. The statuses of these servers are defined by their internal behavior through collaboration request location info and request weather info. The status of the both the servers are defined as idle and processing. When a step (annotated as <<PAstep>> in Figure 4) will be executed the status of the servers will be marked as performance model state from where a new state may be generated with a transition rate or return back to a already marked state with a transition rate mentioned in the annotated UML model in Figure 4. The states of the performance model are shown in Table 1 based on the status of both the servers. The states are: (idle, idle), (processing, idle), (idle, processing), (processing, processing) where the first part defines the status of the location server and second part defines the status of the weather server. If we assume initial marking such as the status of the location server and weather server is idle that means servers have no user request to process then we can derive all the reachable markings to produce reachability graph of the system from
the initial marking according to the arrival or departure of client requests.

**Step 6 – Markov model generation:** If each of the reachable marking is represented as a node and each edge between the nodes level with the trigger of their change by the transition rate (mentioned in annotated UML model by $\lambda$, $\mu_1$, $\mu_2$), we can generate the markov model of the system. We will consider the number of location server and weather server in our system is 3 where each server can process one request at a time. We assume system is stable and both servers buffer capacity are null, the state transition diagram is shown in Figure 9 where system states are generated from location server and weather server status shown in Table 1.

Where idle means no job is in the servers to process and processing means 1 job (number of job in the location server & weather server is mentioned by N & M & here highest value of N=M=1 in Figure 9) is processed by the servers. If we assume the system is stable, both servers buffer capacity is infinite, follow Poisson arrival pattern & FIFO (First In First Out) scheduling policy and servers can process one request at a time, the state transition diagram is shown in Figure 10 (where $N, M > 1$ to infinity) which shows more states than the states generated from the status of both the servers. So if $N=M=1$ then the state transition diagram will be mentioned in Figure 9 which just reflect the internal

### Table 1. States of performance model based on the status of location server & weather server

<table>
<thead>
<tr>
<th>Location Server</th>
<th>Weather Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>Idle</td>
</tr>
<tr>
<td>processing</td>
<td>Idle</td>
</tr>
<tr>
<td>Idle</td>
<td>Processing</td>
</tr>
<tr>
<td>processing</td>
<td>Processing</td>
</tr>
</tbody>
</table>

![Figure 9. State transition diagram of the Markov model when number of request or job arrived and serviced by the system is 1](image-url)

![Figure 10. State transition diagram of the Markov model for 1st variation of deployment diagram](image-url)

![Figure 11. State transition diagram of the Markov model for 2nd variation of deployment diagram](image-url)
behavior of the servers showing the change of system states mentioned in Table 1. If \((N, M) > 1\) the state transition diagram will be mentioned in Figure 10 highlighting more additional states including the states shown in Table 1 & Figure 9 where the states will be marked by the total number of job \(N \& M\) in the server. In this case, if \((N, M) \leq (n, m)\), there are \(N, M\) customers in the system and \(n-N, m-M\) servers are idle and If \((N, M) \geq (n, m)\) there are \(N, M\) customers in the system, the \(n, m\) servers are busy and there are \(N-n, M-m\) customers in the queue (Here \(n, m\) are the number of location servers and weather servers).

The markov model for the second variation of deployment diagram can be generated by following the above procedure as well where application server will be considered as performance limiting factor of the system. The performance model for this case is shown in Figure 11.

**Step 7 – Evaluate model:** The generated performance model will be solved by the SHARPE performance model solver [9] to generate performance prediction result. Some of the performance evaluation results generated by the tool are shown in the graph form in Figure 12 for both variations of the deployment diagrams which highlight their affect on the system performance.

We modeled and solved the system for both design alternatives of system architectures. The performance prediction result such as mean response time and utilization of servers are derived for the same arrival rate of the client requests and service rate of the servers for both design alternatives of the system architecture. The comparison of the result is demonstrated in the graph in Figure 12 for the servers of the both design alternatives which clearly shows how the response time of the system varies with the same server utilization for considering the different system architecture candidates that helps the developers to resolve the bottleneck of system performance by finding a better allocation of service components to the physical components of the system.

**IV. CONCLUSION**

Here our main contribution is delineated as to present the UML collaboration and activity oriented approach to capture the system dynamics that is utilized to sketch the performance model for a distributed system where every collaboration performs separate task. The behavior of the collaboration and the composition of collaboration to highlight the overall system behavior are demonstrated by utilizing UML activity. To present the behavior and composition of the collaboration using activity, we extend the notation to handle the collaboration that is executed not only in the single session but also in multiple sessions at the same time where different instances of collaborations are distinguished and coordinated by adding notation select to our execution profile. The select notation can outline the relations between multiple sessions unambiguously on an abstract level. Later a mapping between collaboration role and system component is outlined to show how the service of the distributed system is realized by the joint behavior of the system components that are physically distributed. Different variations of deployment diagram are considered to generate the performance model to show how the variations in the deployment diagram thus affect the system performance under different work load. Further work includes automating the whole process of translation from our UML specification style to generate a performance model and the way to solve the performance model through our proposed framework as well as to tackle the state explosion problems for large systems.

**REFERENCES**

[1] OMG UML Superstructure, Version-2.2


