A Novel Approach for Performance Improvement of Multi-Agent based System Architecture

Hyunsang Youn  
School of Information and Communication Engineering, Sungkyunkwan University  
wizehack@ece.skku.ac.kr

Suhyeon Jang  
School of Information and Communication Engineering, Sungkyunkwan University  
nickelion@ece.skku.ac.kr

Eunseok Lee  
School of Information and Communication Engineering, Sungkyunkwan University  
eslee@ece.skku.ac.kr

Abstract

In software engineering, it is important issue that architects estimate the performance of software in the early phase of development process. Many approaches translate architecture based on UML into analytical models. However, in agent based system development, these approaches ignore or simplify the crucial details of the underlying the performance of an agent platform. Additionally, architects have only limited kinds of solution to improve performance of system architecture. In this paper, we propose analytical approach for performance improvement of multi-agent based system. This approach avoids the need for a prototype implementation since architects can determine overall form of the performance equation from the architectural design description and can enhance the system architecture to derive optimal architecture from analytical model. We verify the effectiveness of our approach through U-Commerce scenario.

1. Introduction

Performance analysis of early stage of software development has been issued because of verifying non-functional requirement and optimizing software architecture design. Recently, in Object Oriented development, there have been many results that derive analytical model from system architecture designed by UML. Software architects estimate the performance of system architecture using the analytical model. However, in Agent Oriented development, there is a limit to estimate performance of system architecture based on the prior results.

Agents require an agent platform to be configured on a system, or across a distributed network. Agent platform is responsible for the life cycle, security, and behavior of agents running on the platform. The agents use the defined set of platform services for different tasks such as getting a unique name, looking up other agents in a directory, logging, and passing messages to another agent. To estimate the performance of multi-agent based system, we therefore should consider the estimation of the performance of agent platform. Additionally,
we do not need the derivation of performance model from UML based system architecture but also derivation of system architecture from optimized performance model.

In this paper, we propose analytical model based approach for performance improvement of multi-agent based system. This approach avoids the need for a prototype implementation since we can determine overall form of the performance equation from the architectural design description and then, we derive optimal architecture from analytical model.

Analytic model based approach is very useful technique to estimate performance of a system from abstract model. The approach needs some parameter values that is to simulate a model. Analytical model provides mathematical formulas which can represent behavior and performance of system. System designer can predict the behavior and estimate the performance of system without implementation code of the system [11].

Generally, An analytical model for the performance evaluation of software use Queuing Network (QN) based modeling theory such as Extended QN (EQN) or Layered QN (LQN)[2][3][4]. Translation technique to drive QN based model is useful approach to get the analytical from software architecture.

Performance model-based techniques can be used at the early stage of design. The software performance engineering community (SPE) has advocated the need to integrate design and to improve the performance of software for a number of years [7]. The design method is based on use cases, object modeling, and functional modeling using UML. Many approaches translate architecture designs in UML to analytical models, such as queuing network models (QNM) [10]. A survey on model-based performance prediction approaches and support tool can be found in [10]. We use a Queuing Network Model to estimate the performance of Multi-Agent based software.

Cortellessa and Mirandola in [6] propose the methodology, called PRIMA-UML, generates a performance model representing the specified system incrementally from different UML diagram. In PRIMA-UML, Software Architecture is specified by using deployment diagram, Sequence diagram and Use Case diagram. The software execution behavior is derived from the Use Case and Sequence diagrams, and the analytical model of system is derived from the Deployment diagram.

The UML profile for Scheduling, Performance, and Time was described in [8] and has been adopted as an official OMG standard in March 2002. The main aims of the UML Profile for Scheduling, Performance, and Time (Real-time UML standard) are to identify the requirements for enabling performance and methods to model physical time, timing specifications, timing services and logical and physical resources, concurrence and scheduling, software and hardware infrastructure, and their mapping [1].

Many approaches translate the architecture that is represented by UML into analytical models. An application workflow is presented in sequence or activity diagrams. Deployment diagrams are used to describe the H/W and S/W resource. However, these approaches ignore or simplify the crucial details of the underlying the performance of an agent platform. Communication performance between Agent and agent platform is not explicitly modeled. The activities of agent platform are not also explicitly modeled. So the analytical models are rather inaccurate. Another problem of these approaches is design improvement from bottleneck or critical component. In transformation approach from UML to QNM, architects have only several kinds of solution to improve the performance of system architecture. To determine optimal architecture, architects need to do not modify in UML domain but to alter
performance model. In this paper, we therefore propose two-way transformation approach from architectural model to performance model and vice versa.

The remainder of this paper is organized as follows: In Section 2, we describe the approach to estimate the performance of multi-agent based system. In Section 3, we explore the approach by using scenario based on ubiquitous commerce system and we verify the usefulness of our approach. Finally, we conclude in Section 4 with an overview of our plans to further evolve this work.

2. Proposed Approach

In this section, we propose an analytical approach for performance improvement of multi-agent based software. To analyze and improve the performance of a given application architecture, firstly, we define multi-agent based system architecture noted by UML and performance model based on queuing network. Secondly, we propose two-way transformation approach from system architecture based on UML to queuing network model, and vice versa.

We describe structural aspect and behavioral aspect to represent the architecture of multi-agent based system with agent platform. The structural aspect consists of platform independent structure and platform dependent structure. To illustrate platform independent structure, we use the subset of class diagram. We can identify associating and multiplexing of each entity from platform independent structure. A platform dependent structure is used to represent H/W and agent platform to run multi-agent based software. The platform dependent structure is illustrated by the subset of deployment diagram. In behavioral aspect, we describe the external behavior of the architecture of multi-agent based software by using the subset of message sequence diagram. We use the subset of activity diagram to represent the internal behavior of each agent. To get the parameters of queuing network, we add SPT profile [9] to internal and external behavior model. We illustrate architectural model and performance model as follows:

\[
\text{ArchitecturalModel} = A(x) \\
\text{QueuingNetworkModel} = Q(x)
\]

\[
A(X) = \{A_{PIS}(x), A_{PDS}(x), A_{EB}(x), A_{IB}(x)\} \\
Q(X) = \{Q_{App}(x), Q_{Platform}(x)\}
\]

\(A(X)\) is architectural model generation function based on UML. The function has four sub-functions; platform independent structure, platform dependent structure, External behavior, Internal behavior. \(Q(X)\) is generation function for queuing network model(QNM). The function has two sub-functions; Application QNM generation function, Platform QNM generation function. We define input parameters to the functions for Architectural Model. The function has four sub-functions; platform independent structure(\(A_{PIS}(x)\)), platform dependent structure(\(A_{PDS}(x)\)), External behavior(\(A_{EB}(x)\)), Internal behavior(\(A_{IB}(x)\)). The function has two sub-functions; Application QNM generation function, Platform QNM generation function. We define input parameters of above four functions for Architectural Model.

\[
x_{PIS} = \{\\text{User}, \text{Order}, \text{Sale}, \text{SaleN}, \text{CashN}, \text{Cash} \}
\]

\[
x_{PDS} = \{\\text{User}, \text{Order}, \text{Sale}, \text{SaleN}, \text{CashN}, \text{Cash} \}
\]

\[
x_{EB} = \{\\text{User}, \text{Cash}, \text{Sale}, \text{SaleN}, \text{CashN}, \text{Cash} \}
\]

\[
x_{IB} = \{\\text{User}, \text{Cash}, \text{Sale}, \text{SaleN}, \text{CashN}, \text{Cash} \}
\]
The input parameters for APIS(x) is set of <<agent>>, <<object>>, association, multiplex, role name, association name and dependency. The input parameters for APDS(x) is set of <<agent>>, <<object>>, <<device>>, <<communicator>>, <<platform>> and dependency. The input parameters for AEB(x) is set of <<agent>>, <<object>>, message, alternative, parallel, <<PA_CloseLoad>> and <<PA_OpenLoad>>. <<PA_CloseLoad>> and <<PA_OpenLoad>> are defined in UML SPT profile. To represent MAS architecture, we add UML stereotypes, which are <<agent>>, <<object>>, <<device>>, <<communicator>> and <<platform>>.

We also define following input parameters for QNM generation function.

\[ X_{queueing} = \{ \lambda, \mu, p, t, AP \} \]

\( \lambda \) is arrival rate for open queueing network. \( \mu \) is service rate for server. \( p \) is routing probability from server to queue. AP is open queueing network model which represents agent platform. The AP is associated with queueing network for application level.

A performance model should explicitly represent a behavior of agent platform. FIPA (Foundation of Intelligent Physical Agent) standard specification shows Behaviors of Agent Platform for Message Transport. We create a QNM for the agent platform based on FIPA standard specification. We abstract the details of each module and their communication which consist of the platform, resulting in a model with a simple structure. Standard queuing theory techniques enable us to obtain a performance measures from the model, once we determine the service demand distribution characteristics of the queues in the model. In Fig.1, we define transmission according to external, internal and local. External transmission illustrates that agents send their message to other agents which are running on other platform. Internal transmission describes that agents send their message to other agents that are running on same platform, but not same host (or server). Local Transmission is shows agents send their message to other of the standard agent platform and its running H/W environment; such as encoding/decoding mechanism, network bandwidth, etc. They reflect the internally hidden implementation of the platform. We therefore implement a simple application and measure its performance. The logic of our application is very simple. Our application inputs any typed parameters into the module that we want to earn operating time. And then, we get the time that is measured by difference between end time and start time. We combine the queuing network of application domain with that of agent platform. The QNM of application can be defined either open queuing network or closed queuing network by annotating SPT profile. Some of the parameters represent tunable features.
To derive QNM from Architectural Model based on UML and vice versa, we define transformation function as follows.

\[
\text{Trans}(X_{arch}) = X_{queueing} \\
\text{Inverse}(X_{queueing}) = X_{arch}
\]

First, we define the sub-functions of transformation function listed in Table 1. In this table, \(X_{queueing}\) is the elements of queuing network model, and \(X_{arch}\) is the element of multi-agent system architecture to represent the architecture based on UML.

<table>
<thead>
<tr>
<th>QNM (X_{queueing})</th>
<th>System Architecture (X_{arch})</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue</td>
<td>Agent, Object</td>
<td>(T_q(x) \iff I_q(x))</td>
</tr>
<tr>
<td>Number of Server</td>
<td>Multiplicity, Par</td>
<td>(T_{n}(x) \iff I_{n}(x))</td>
</tr>
<tr>
<td>Routing Probability</td>
<td>Alt, Decision, Performance Profile</td>
<td>(T_{rp}(x) \iff I_{rp}(x))</td>
</tr>
<tr>
<td>Arrival Rate</td>
<td>Performance Profile</td>
<td>(T_{ar}(x) \iff I_{ar}(x))</td>
</tr>
<tr>
<td>Service Rate</td>
<td>Multiplicity, Performance Profile</td>
<td>(T_{sr}(x) \iff I_{sr}(x))</td>
</tr>
<tr>
<td>NOC*</td>
<td>Performance Profile</td>
<td>(T_{noc}(x) \iff I_{noc}(x))</td>
</tr>
<tr>
<td>RP** of Agent Platform</td>
<td>Agent, Dependency</td>
<td>(T_{ap}(x) \iff I_{ap}(x))</td>
</tr>
</tbody>
</table>

\(\ast\)NOC is the number of customers. **RP is Routing Probability.

\(T_q(x)\) is the function to identify ‘queue’ that is the element of QNM from the system architecture, \(T_q(x)\) extract agent or object from platform dependent structure, platform independent structure, and external behavior. \(I_q(x)\) performs the discovery of agents or objects from QNM.

The number of server for M/M/n queuing network is determined by multiplicity. To identify the number of server connecting with one queue, \(T_s(x)\) recognize ‘par’ combined fragment of external behavior. The number of multiplexing is equal to the number of server. \(I_s(x)\) returns ‘par’ combined fragment from number of server in queuing network.
In Tar(x), arrival rate for QNM is described by ‘PAoccurrence’ tag at SPT profile. Iar(x) returns ‘PAoccurrence’ from arrival rate described in internal behavior.

Tnoc(x) extracts the number of customer for QNM is determined by ‘PApopulation’ tag at SPT profile. Inc(x) annotates ‘PApopulation’ tag from the number of customer in QNM.

Service rate for QNM is determined by the number of server. Once an agent multicasts a message to other agents, Agent platform copies the message equal to the number of agent that receives it. However, the scheme effect on waiting time of jobs in queue. We therefore should tune service rate of server according to the number of server. In Tsr(x), x = \{T, k\}, k is the number of multicasting, processing time of an agent, k is, So we can define the service rate \( \mu_k \) that the number of agent that is to multicast a message is k.

\[
\mu_k = Tsr(T, k)
\]

Let’s assume that T1 is the service time when only one server process jobs in one queue.

\[
T2 = 2T1 + \alpha
\]

\( \alpha \) is the copying time of message that is to multicast. However, this value is near upon 0.

\[
\begin{align*}
T_k &= k \cdot T1 \\
\mu &= \frac{1}{T} \\
\therefore \mu_k &= \frac{1}{T_k}
\end{align*}
\]

In inverse function:

\[
\begin{align*}
I_{sr}(x), x = \{\mu_k, k\}, T = I_{sr}(x) \\
k \cdot T = \frac{1}{\mu_k} \\
\therefore T = \frac{1}{\mu_k}
\end{align*}
\]

Once one agent sends a message to another agent, Routing Probability of agent platform is determined by platform dependent structure. Each server of one queue can have their routing probability to send a job to one queue of agent platform.

Tap(x) = X = \{X \mid X = \text{Routing Probability of Agent Platform}\}

Marking of routing probability X: X = \{External, Internal, Local\} Let’s assume that:

Case 1 is that agent1 and agent2 which are communicating is not running on same platform but also same device.

Case 2 is that agent1 and agent2 which are communicating is running on same platform but different device.

Case 3 is that agent1 and agent2 which are communicating is running on same device.

In case 1: X = \{1, 0, 0\}

In case 2: X = \{0, 1, 0\}

In case 3: X = \{0, 0, 1\}
Iap(x) identifies current case from Marking of routing probability X. Therefore, we can find Iap(x) using above marking.

3. Evaluation: CASE STUDY

In this section, we verify the usefulness of our approach through following scenario which extends mobile commerce system [12]. Through the scenario, we deliver the architectural specification of multi-agent based system and transform the architecture into queuing network model.

Scenario: A clerk explains some products to customers. The customers use RFID reader attached to their mobile phone to get information of the product. RFID badge attached on the product give the phone link. The customers connect to online or offline shop though the link; the offline shop is in around the customer. If there is a shop to buy the product, whether offline or online, as the lowest price, the customers buy the product using their mobile phone.

![Fig 2. UML based Architectural Model of U-Commerce System](image)

Figure 2 show multi-agent based system architecture represented by UML; platform independent structure (class diagram), platform dependent structure (deployment diagram), external behavior (sequence diagram), internal behavior (activity diagram).

The platform independent structure shows the relationship of each agent. Platform dependent structure show communication and operation environment of multi-agent. For example, Although Offline-Shop agent and Online-Shop agent run on same agent platform, they are operated on different device. User agent and Online-Shop agent does not run on same agent platform, but also they are operated by same device.
To represent external behavior of agent, there are many scenarios. In this paper, we choose one scenario, which explains the ordering product and the payment of the system. In internal and external behavior, we add the annotation that is noted by SPT profile to describe some parameters and limitations that are used to derive queuing network. In internal behavior, to define service time of each server of queuing network model, we annotate SPT profile to each activity that describe task of an agent. We benchmark the M-commerce system which has developed before to extract the parameter values that represent ‘PAdemand’ of SPT profile. We can configure service time of each server of queuing network from ‘PAdemand’. For example, in Fig.2, User Agent performs ‘Sorting’ activity and then performs ‘Recommend’ activity in a parallel operation with ‘Display’ activity. The ‘Sorting’ activity spends 1.5ms, ‘Recommend’ activity and ‘Display’ activity spend 1ms. So overall spend time is 3.5ms.

**Fig 3. Queuing Network Model of U-Commerce System**

Figure 3 shows the queuing network model derived by system architecture. We simulate the model, and detect potential bottleneck of the system. To resolve the potential bottleneck, we modify the model in queuing network domain. The modified model recommends alternative system architecture. Figure 4 shows the modified queuing network model and the altered platform dependent structure of system architecture. From new architecture, we can modify the previous scenario. In new scenario, we consider that the shopping mall server of offline-shop closed to user is on same platform.

**Fig. 4. Queuing Network Model of U-Commerce System**
4. Conclusion

In this paper, we reported a contribution to the area of performance improvement for multi-agent based software. We proposed a two-way transformation approach from architectural model to performance model and vice versa. The transformation approach reflected the crucial details of the underlying the performance of an agent platform. Additional advantage of this approach is that an analytical model of agent platform which is reusable. Our approach derives a queuing network model for the architecture with parameters that reflect performance properties of the platform. The parameters can be measured by our simple application externally. We verified the effectiveness of our approach through U-Commerce scenario.

5. References

Authors

**Hyunsang Youn** is currently pursuing the PhD degree at Sungkyunkwan University. received M.S. degree in computer engineering at Sungkyunkwan University, Korea, in 2006. He obtained the B.S. degree in Information Communication Engineering at Seokyeong University, Korea, in 2004, He is a member of the Software Engineering Lab. His research interests the area of model based software performance prediction and analysis, Autonomic Computing System for Self-management, Model Driven Architecture. Currently, He works for intelligent context aware agent platform development in the Ubiquitous Computing Research Institute.

**Suhyeon Jang** received the MSc degree in computer engineering at the Sungkyunkwan University, South Korea, in 2007. He previously worked for intelligent context aware agent platform development in the Ubiquitous Computing Research Institute. His research interests include multi-agent development methodology base on MDA and software reverse engineering. Currently, he is a software developer in NHN Corporation.

**Eunseok Lee** received the Ph.D. and M.S. degrees in Department of Information Engineering at Tohoku University, Japan, in 1992 and 1988, respectively, and his B.S. degree in Electronic Engineering at Sungkyunkwan University, Korea, in 1985. He is a Professor of the Department of Computer Engineering of Sungkyunkwan University, Korea. From 1994 to 1995, he was an assistance professor of the Department of In-formation Engineering of Tohoku University, Japan. He was a research scientist in In-formation and Electronics Laboratory of Mitsubishi Electric Corporation, Japan, from 1992 to 1994. His areas of research include Software Engineering, Autonomic/Ubiquitous Computing and Agent-oriented Intelligence System.