International Journal on Perceptive and Cognitive Computing (IJPCC) Vol 10, Issue 1 (2024)
https://doi.org/10.31436/ijpcc.v10i1.429

Multi-Agent System in Web Services

Najhan Muhamad Ibrahim[1*], Mohd Fadzil Hassan[2], Muhammad Amrullah BIN DRS Nasrul[3]
[1]Department of Information Systems, Kulliyyah of Information and Communication Technology,
[2]Department of Computer and Information Sciences Universiti Teknologi PETRONAS, Bandar Seri Iskandar, 32610, Perak, Malaysia.
[3]Ahmad Ibrahim Kuliyyah of Laws, International Islamic University Malaysia, Gombak, Selangor, Malaysia.

*Corresponding author: najhan_ibrahim@iium.edu.my
(Received: 27th September 2023; Accepted: 1st December 2023; Published on-line: 28th January 2024)

Abstract— The term "service-oriented architecture" (SOA) refers to a software paradigm for creating systems made up of a variety of services that interact with one another to accomplish a certain task. The communications involve more than just transmitting data back and forth; they also connect two or more services to coordinate the required operations. Cross-platform communication is necessary for the coordinating process when services are spread across several platforms. Several web service standards and specifications are used in the present SOA implementation. Based on the prototype's implementation and simulation, this proposed research study has been validated and evaluated. The suggested cross-platform communications architecture is implemented using NetBeans, JADE, WSIG, and OWL-S. An integrated development environment for Java is offered by Oracle in the form of NetBeans. Telecom Italia's Java Agent DEvelopment Framework (JADE) is an agent software framework that is entirely built in the Java language. Web Service Integration Gateway (WSIG), which helps to facilitate JADE agent services being called by Web service clients. The core fundamental engine of the suggested framework is OWL-S. It is a web ontology language used to describe Semantic Web Services in the Semantic Web's OWL-based framework. A quantitative approach is used in this study's performance analysis and comparative investigation for evaluation and validation. The prototype's major element is the Java agent development framework (JADE), which was used to create a multi-agent system for the agent-based MOM framework that has been presented. The creation of the multi-agent systems was facilitated by the JADE platform. JADE 3.7 was the version that was employed. Two key features of JADE are a FIPA-compliant agent platform and a package for creating Java agents. The implementation results show that the proposed agent-based MOM framework was successful communicate between multiple types of SOA application with a better performance of the average of round-trip time where the proposed framework was successful in responding to all the requests.

Keywords— Service Oriented Architecture (SOA), Web services, Java Agent Development Framework (JADE), Web Service Integration Gateway (WSIG), Multi-agent system (MAS).

I. Introduction

The Java runtime environment, version 1.5, is the typical system requirement for Java Agent Development Framework (JADE), and it is available for free download from Sun Corporation [1]. Java Agent Development Framework (JADE) is entirely implemented in the Java language. In accordance with the agents' specification, each agent performs a variety of actions. An agent platform that complies with FIPA and a package for creating Java agents are JADE's two key features. It is made up of several Java packages that offer application programmers ready-built functionalities as well as abstract interfaces for creating original apps [2]. A JADE agent can be invoked with the help of the add-on component known as Web Service Integration Gateway, which also gives the web service and the agent software a place to integrate. Through the usage of WSIG, services offered by agents and made available in the JADE DF library can easily be exposed as web services and vice versa. In order to address certain requirements, it gives developers flexibility and liberty. The servlet and agent that make up WSIG are its two key components. The web service message is served by the WSIG servlet, which also extracts it, prepares the associated agent action, and passes it to the WSIG agent. The WSIG servlet is the front-end to the internet world. WSIG agent is the gateway between the application and agent world and is responsible for forwarding agent actions from the WSIG servlet to serve them and getting back respond [3]

Moreover, the web service and agent software are supported in their integration and communication by WSIG. According to Figure 1, WSIG is depicted as a web component with the two primary attributes WSIG Servlet and WSIG agent. Incoming web service requests are handled by the WSIG Servlet, which also extracts the message and serves as an interface to other programmes. In order to prepare the client response, it prepares the necessary agent action and sends it to the WSIG agent. However, the WSIG agent can be thought of as the intermediary between the web and agent software. In addition to subscribing to the JADE DF to receive notifications, it is in charge of relaying agent actions that are received from the WSIG Servlet. The WSDL for each agent service registered with DF was produced for the purpose of agent registration and deregistration, and the service was then made available in a UDDI registry [4,5].

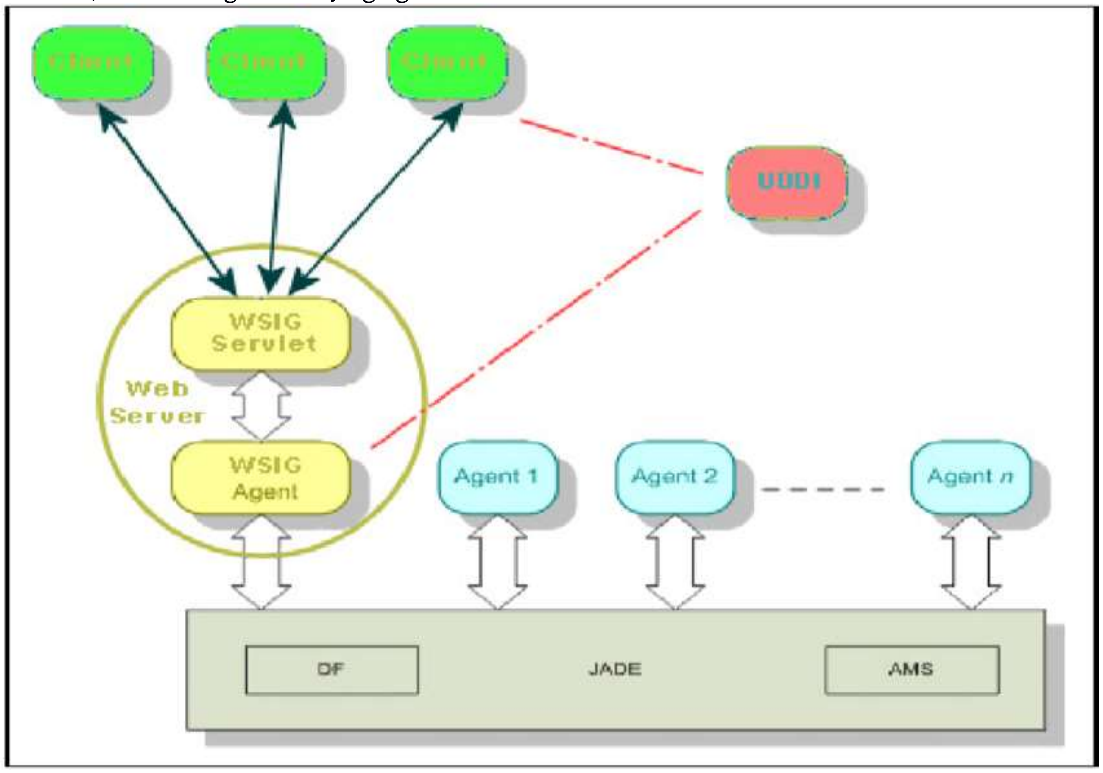


Fig.1. WSIG Environment (Board, 2008)

An ontology for web services called OWL-S adds a core set of markup language structures for describing the features and capabilities of their web services in a clear and understandable way for computers. The OWL-S markup of web services will make it easier to automate operations including the discovery, execution, composition, and interoperation of web services. The WSDL is now expanding quickly to serve as a foundation for web service interoperability. OWL-S is therefore being developed to offer integration between them and the agent technology. It is the adaptable automation of the supply of services and it provides important techniques [6]. Also, For describing semantic web services, OWL-S is the ontology of the OWL-based framework in the Semantic Web. It will make it possible for users and software agents to automatically translate, discover, invoke, compose, and monitor online pages that deliver services within certain parameters.

II. Experimental Setup

The core element of the prototype is the Java agent development framework (JADE), which was utilised to create a multi-agent system for the agent-based MOM framework that has been suggested. The suggested multi-agent system's roles are described. In the suggested prototype, there are two key plugin elements. The web service ontology, or OWL-S, is the first. It is used to describe semantic web services and is the ontology of the Semantic Web's OWL-based framework. Adaptive web service discovery, execution, composition, and interoperation are just a few of the chores that can be automated thanks to this. To describe web services in a uniform manner, WSIG also includes WSDL. For publishing web services using tModels, it supports UDDI repositories and SOAP/HTTP messages for transmission. In the Universal Description, Discovery, and Integration (UDDI) registry, a service type is represented by a data structure called a tModel, which is a general representation of a registered service. Additionally, WSIG supports the web service description process. The procedure, which is registered with the DF, entails creating appropriate WSDL for each web service description. The exposed service is then optionally published in a UDDI register. Additionally, it makes it possible to map the Agent Communication Language (ACL) into WSDL and the other way around [2, 7].

One of significant research works that focuses on interoperability communications framework, which models and discusses the requirement attributes for the collaboration between different application systems. Refer to [8] strongly argued that the development of interoperability communications framework to support communications among diverse and geographically distributed system are significantly important. They have proposed six main requirement attribute with twenty-seven sub-attributes, which is a general requirement for distributed and collaboration application. In the sub-requirements, several significant requirements for interoperability communications have been highlighted. Nevertheless, the main proposed requirements for interoperability communications in this research work are too general to be incorporated in the concrete interoperability communications framework. This research work is most likely focus on non-functional requirement attributes. Therefore, the comprehensive literature study need to be conducted to evaluate and consolidate others requirement attribute to be included in the proposed interoperability communications [9].

A topology with three different SOA-based applications was constructed, as shown in Figure 2, in order to test the prototype of the suggested agent-based MOM framework. The prototype's simulation is built up in two distinct scenarios: one to test the multiplicity of cross-platform communications and the other to validate cross-platform communications across several platforms. Regarding the validation method, the simulation configuration for each scenario differs. First, the proposed agent-based MOM framework was simulated in terms of its cross-platform communications capacity, and two separate SOA-based applications were simulated. This simulation was done to validate the cross-platform communications. The simulation of three various types of SOA-based applications that are attached to an agent-based MOM framework serves as the second step in validating the diversity of cross-platform communications. Measurement of the round-trip time for each of the created requests was one of the evaluation metrics. These requests were put to the test at two different runtime levels, namely the low level and the high level [10].

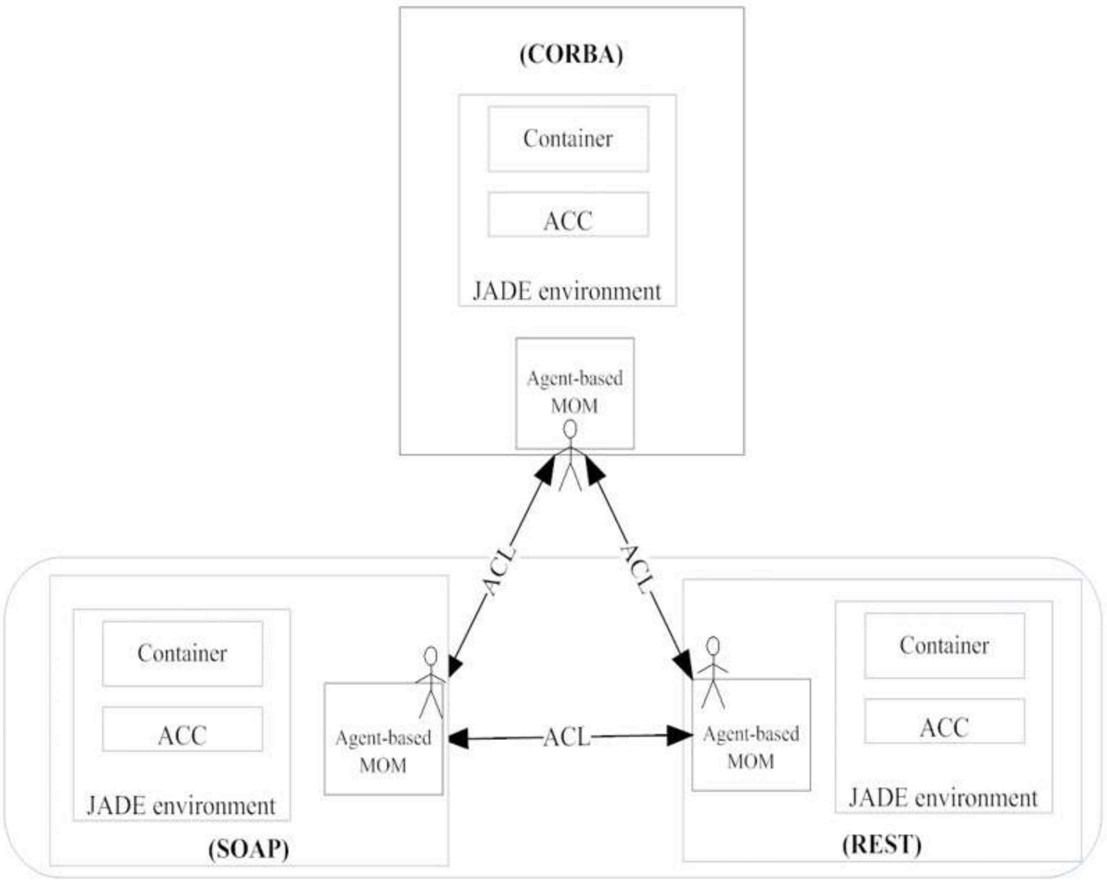


Fig. 2. Simulation Setup

Furthermore, the variation of the simulations measured by the communications between multi-agents system that analyzed at the source code level of the agent. Several directions of the cross-platform communications were classified that were used for evaluation and validation purposes in this research work. The java method was used (long System.currentTimeSeconds()) to measure the time intervals that returned the number of seconds as a standard agent-based measurement [5]. The simulation data of the proposed agent-based MOM framework in comparison to existing MOM and JMS to validate the performance of the cross-platform communications and multiplicity cross-platform communications.

Table I
Simulation Parameters

Framework	**Traffic**	**Total Number of Requests (at run time)**
Agent-based MOM Framework (SOAP - REST)	Low Level	{10, 20, 30, 40, 50}
Agent-based MOM Framework (SOAP – REST)	High Level	{60, 70, 80, 90, 100}

The simulation's primary goal is to determine whether the suggested framework achieves its desired goals. The experiments are designed to assess the effectiveness of the suggested agent-based MOM framework by measuring the communications round-trip (request/respond) time in seconds in comparison to traffic as represented by the volume of requests initiated during the run time. The common metric for assessing the effectiveness of communications in MOM and multi-agent systems is round-trip time. The number of requests is constrained by the JADE platform's limited capability for devices other than personal computers. The low level of the request is 10, 20, 30, 40 and 50 requests allocated and the high level of the request is 60, 70, 80, 90 and 100 requests allocated at run time as presented in Table 1. Similarly, Table 2 represented the simulation parameters to validate the multiplicity of cross-platform communications. To validate the multiplicity of cross-platform communications, three different types of SOA application were simulated as presented in Figure 2. The communications round-trip time (request/ respond) against the request at the low level and the high level of traffic will also be evaluated. The simulation metrics are chosen in such a way that they are sufficient enough to execute the request in cross-platform environments. These simulation metrics were not altered for both validations performed in this research because the objectives of the research are to validate the cross-platform communications and the multiplicity of the cross-platform communications.

Table II
Simulation Parameters for Multiplicity

Framework	Traffic	Total Number of Requests (at run time)
Agent-based MOM Framework (SOAP – REST –CORBA)	Low level	{10, 20, 30, 40, 50}
Agent-based MOM Framework (SOAP – REST –CORBA)	High Level	{60, 70, 80, 90, 100}

As shown in Figure 2, agent-based MOM framework resides in the SOAP-based application, CORBA-based application, and REST-based application. All applications run in JADE environment. Three essential elements are presented in the FIPA compliant platform, which is Agent management service (AMS), Directory facility (DF) and Agent communication channel (ACC). AMS controls the access of the platform and DF. DF provides a library service and ACC that facilitates the message transport service for FIPA ACL message delivery among agents living in different agent platforms [1]. The communications between SOAP-based and REST-based application were simulated to validate the cross-platform communications. SOAP-based represented the SOA application that using SOAP web service message and REST-based represented the SOA application that using REST web services message. On the other hand, the communications between SOAP-based application, REST-based application and CORBA-based application were simulated to validate and evaluate the performance of multiplicity cross-platform communications.

III. SIMULATION APPROACH

As shown in Figure 3, the same topology was used for both validate of cross-platform communications and validation of multiplicity cross-platform communications. For the validation of the cross-platform communications, SOAP-based and REST-based application were simulated.

Invalidations of the multiplicity of cross-platform communications, all three SOA-based applications were simulated. Several requests from SOAP-based to REST-based application were sent to simulate the proposed cross-platform communications and also to CORBA-based application to simulate the multiplicity of cross-platform communications. The round trip time of the request to the response was used to analyze the performance and comparative studies. The simulation to validate the cross-platform communications was compared with existing MOM. In this simulation, the proposed agent-based MOM framework was simulated with two different types of SOA application as supported by the existing MOM. Due to the similar research work that supports multiple types of SOA applications in runtime is not available, the simulation to validate the multiplicity of cross-platform communications will be compared with java message services (JMS). JMS is the standard message-oriented middleware by Sun Microsystem, which provides similar functionality with the proposed cross-platform solution and was developed based on the Java environment. However, JMS functionality are incapable of supporting multiple types, and directly respond during runtime as the proposed framework, which capable of setting each agent to respond based on the request [13-15].

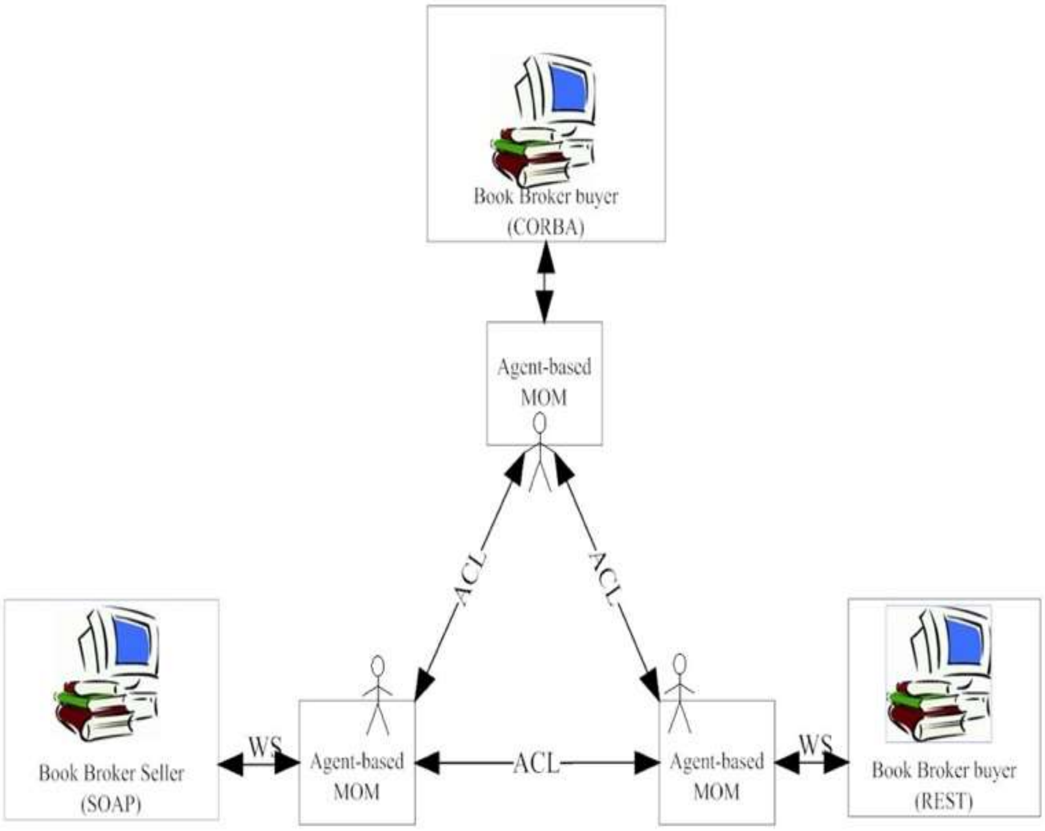


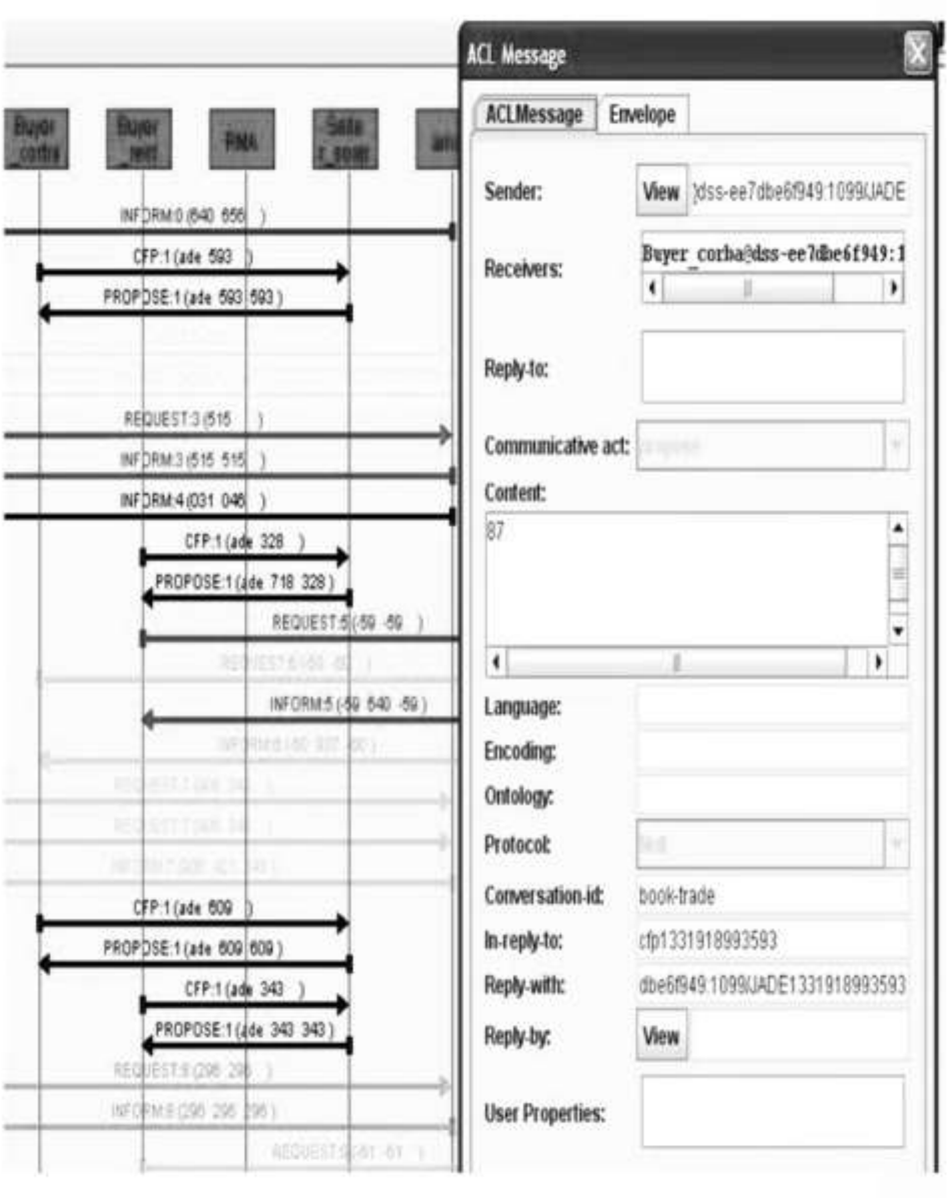
Fig. 3. Simulation Topology

As shown in Figure 3, SOAP-based application represents book broker for the seller and the two book REST-based and CORBA-based applications represent brokers as buyers. Each of SOA-based application is capable of performing as PABMOM and CABMOM in the simulation. For instance, the PABMOM sends the message based on SOAP, which will be de-scripted into WSDL format. Then the message is translated into ACL (Agent Communication Language) to enable the communications between multi-agent systems. The soap-based application can also act as the CABMOM to buy the book offered by other applications. Two significant situations are to activate the communications process, which is the PABMOM offers to sell the book and the CABMOM requests to buy the book.

In the case of book broker buyer requested to buy the book, it will send a request message to the seller PABMOM. The message includes specific information, such as the book title, max cost and the time of the request. If the book title is available in the current offer then the seller PABMOM will check whether the max cost of the request is in the range of the best price and min price from the offer. In the case, the book title matched but the requested price is not in the range, then the seller PABMOM will send a denial message of the transaction to the buyer CABMOM. On the other hand, if the book title matched with the current offer and the requested price is in range, the confirmation of selling will be sent instead of the denial message. The detail of how the translator process and translation mapping work. The following describes in details the functions used in the selling process (PABMOM) and the buying process (CABMOM) of the simulation.

A. Selling Process (PABMOM)

The PABMOM has a GUI by means which the user can insert new book titles and the associated prices in the local catalog of the books for sale. The PABMOM also continuously waits for a request from a CABMOM. Once the PABMOM receives a request to provide an offer of the product, the agent receiver will check whether the requested book is in the catalog and will reply with the price. Otherwise, the agent will refuse the request. Figure 4 shows the actual transaction of the PABMOM in which Buyer_corba@dss is the CABMOM side. The CABMOM appoints the agent receiver in the translation model. Furthermore, each message transmission describes its activities, i.e., INFORM, CFP (Call for proposal), PROPOSE and REQUEST. INFORM is used to inform about the offer by PABMOM. CFP is used to call for proposal from CABMOM regarding the book to sell. PROPOSE is used to propose the book to sell and REQUEST is used to query the book to sell from PABMOM.



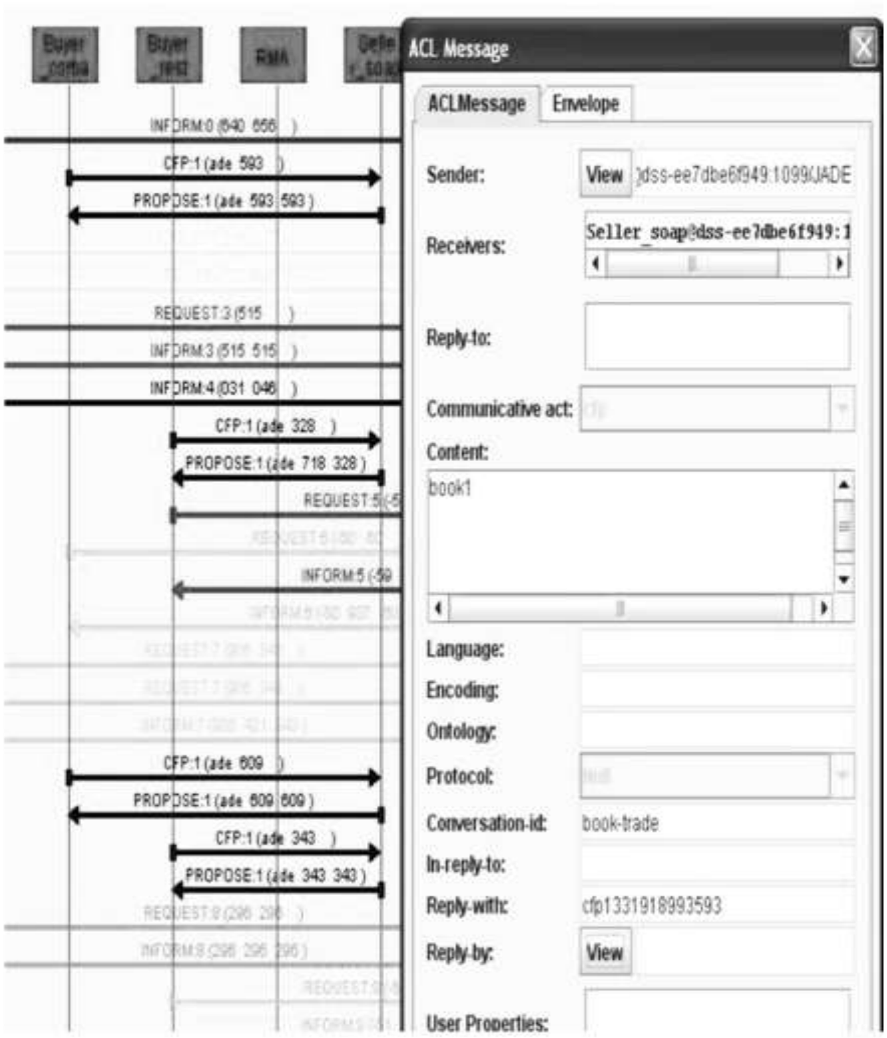
Sender:)dss-ee7dbe6f949:1099/JADE
Receivers: Buyer_corba@dss-ee7dbe6f949:1
Reply-to:
Communicative act:
Content: 87
Language:
Encoding:
Ontology:
Protocol:
Conversation-id: book-trade
In-reply-to: cfp1331918993593
Reply-with: dbe6f949:1099/JADE1331918993593
Reply-by:
User Properties:

Fig. 4. ACL Message from PABMOM to CABMOM

In addition, the request from the CABMOM can be a request to query an offer for a book to sell. A possible solution to achieve that is to make the PABMOM execute two cyclic behaviors: one is dedicated to serving requests for offers, and the other is dedicated to helping purchase orders. Therefore, this is how incoming requests from CABMOM are received. Moreover, it is necessary to make the PABMOM execute a one-time behavior such as updating the catalog of the products available for sale whenever the user adds a new book to sell from the GUI.

B. Buying Process (CABMOM)

CABMOM will be receiving the title of the books for sale (the product to sell) as a command line argument, and sometimes CABMOM is requesting the PABMOM to provide the offered product. When the offer is received, the CABMOM accepts it and issues a purchase order. In case of more than one proposals are provided by the PABMOM, the CABMOM agrees with the lowest price, where price is the element to analyze before accept or reject the offer. After buying the book, the CABMOM terminates the offer. In Figure 5, the actual CABMOM transaction is shown in which the receiver agent is Seller_soap@dss in PABMOM side. The content of the message is the title of the book to buy.



Sender:)dss-ee7dbe6f949:1099/JADE
Receivers: Seller_soap@dss-ee7dbe6f949:1
Reply-to:
Communicative act: cfp
Content: book1
Language:
Encoding:
Ontology:
Protocol: Null
Conversation-id: book-trade
In-reply-to:
Reply-with: cfp1331918993593
Reply-by:
User Properties:

Fig. 5. ACL Message from CABMOM to PABMOM

In addition, the CABMOM uses the CFP (Call for proposal) method to send the message that the CABMOM query an offer of a product from the PABMOM. The content of the CFP message with the book title and price is as shown in Figure 6. The CFP method starts by creating a new CFP object with new ACLMessage(ACLmessage.CFP);. Then the CABMOM searches for all the agent sellers, which provide the offer. The target of CFP message is book title and price. The PROPOSE method is used in CABMOM for the message carrying of the CABMOM to request a book, and the ACCEPT_PROPOSAL method is used for message carrying to accept the offer. Finally, the REFUSE method is used to refuse a proposal from the PABMOM.

```
  // Message carrying a request for offer
ACLMessage cfp = new
ACLMessage(ACLMessage.CFP);
for (int i = 0; i < sellerAgents.lenght; ++i) {
cfp.addReceiver(sellerAgents[i]);
}
cfp.setContent(targetBookTitle&Price);
    myAgent.send(cfp);
```

Fig. 6. Call for Proposal (CFP) Message

IV. THE FINDING

To evaluate and validate the proposed agent-based MOM framework, the communications between multi-agent systems in the experiments are recorded in two different levels. The first level is observing the communications at the code level by observing the performances in the source code. The second level is observing the communications in the application level, i.e., observe the communications process and round-trip time performance between applications. Additionally, all the experiments scenarios have been implemented in two cases. Namely, the simulation to validate the cross-platform communications and the simulation to validate the multiplicity of cross-platform communications at the low level and the high level of the request as presented in Table 3 and Table 4 respectively. It is to check the actual performance between the proposed agent-based MOM framework and existing MOM. SOAP-based and REST-based SOA application is simulated to validate the cross-platform communications, and SOAP-based, REST-based, CORBA-based SOA application is simulated to verify the multiplicity of cross-platform communications. The low level of request execution means that 10, 20, 30, 40 and 50 requests were allocated at the run time. The high level of request execution means that 60, 70, 80, 90 and 100 requests were allocated at run time during execution of the simulation. The low level and the high level of traffic were defined based on the availability metric, which the value of total available time to process the request and the value of outages during low level and high level of request for agent-based MOM framework and existing MOM are different. All these simulations were repeated 20 times to increase the accuracy of the simulation results.

Table 3
Simulation Parameters for Cross-platform Communications (low level)

Framework	Traffic	Number of Requests (at run time)
Agent-based MOM Framework (SOAP - REST)	Low Level	{10, 20, 30, 40, 50}
Existing MOM (SOAP – REST)	Low Level	{10, 20, 30, 40, 50}

Furthermore, the proposed agent-based MOM framework examined the communications to validate the performance of cross-platform communications. To evaluate the communications of the proposed framework, the measured values of the simulation metrics were observed based on twenty different execution times of communications scenario, twenty times in the low level of the request and twenty times in the high level of traffic allocated during runtime. It is to find an accurate average result of each simulation metrics. Then, the measured values were compared with the values obtained from existing MOM and JMS to evaluate the cross-platform communications and multiplicity cross-platform communications respectively. The simulation metric that was measured in the experiments were average of the round-trip time, availability and scalability will be evaluated.

Table 4:
Simulation Parameters for Cross-platform Communications (high level)

Framework	Traffic	Number of Request (at run time)
Agent-based MOM Framework (SOAP – REST)	High level	{60, 70, 80, 90, 100}
Existing MOM (SOAP – REST)	High Level	{60, 70, 80, 90, 100}

Table 5 presents the results of the average round-trip time experiment for the proposed agent-based MOM framework and the existing MOM. The results are ranked by the number of requests allocated and categorized by two levels of request during runtime, which are low level and high level of traffic. Therefore, the second and third column of the first row in Table 5 contains the number of requests allocated and the average round-trip time respectively. That mean, the value of 11.3 and 30 for the average round-trip time in the first row of the agent-based MOM framework and the existing MOM are actually the average round-trip time for the first ten requests allocated during runtime.

Table 5 presents the results of the average round-trip time experiment for the proposed agent-based MOM framework and the existing MOM. The results are ranked by the number of requests allocated and categorized by two levels of request during runtime, which are low level and high level of traffic. Therefore, the second and third column of the first row in Table 5 contains the number of requests allocated and the average round-trip time respectively. That mean, the value of 11.3 and 30 for the average round-trip time in the first row of the agent-based MOM framework and the existing MOM are actually the average round-trip time for the first ten requests allocated during runtime.

Table 5:
Average round-trip time for Agent-based MOM framework and Existing MOM (low level)

Framework	No. of requests at low level	Average Round-trip time (sec)
Agent-based MOM Framework (SOAP – REST)	10	11.3
	20	20
	30	31.3
	40	39.5
	50	48.2
Existing MOM (SOAP – REST)	10	30
	20	42
	30	62
	40	80
	50	97

Likewise, in the third column of Table 5 also contains the average of round-trip time but in the high level of the request. The average round-trip time in the first row of the agent-based MOM framework and the existing MOM is the average round-trip time for the 60 requests allocated during runtime. As seen in Table 4 and Table 5 the value of average round-trip time of the proposed agent-based MOM framework and the existing MOM are increasing linearly due to the increasing number of requests, which required more time and computing resources to process all the requests. The results in Table 4 and Table 5 also show that huge different of round-trip performance between the proposed agent-based MOM framework and the existing MOM at the low level and high level of traffic where the average round-trip time of exiting MOM are double compared to the proposed agent-based MOM framework. The results also show that the average round-trip time of the proposed agent-based MOM framework did show lower increased of the average round-trip time while increasing the number of requests in runtime.

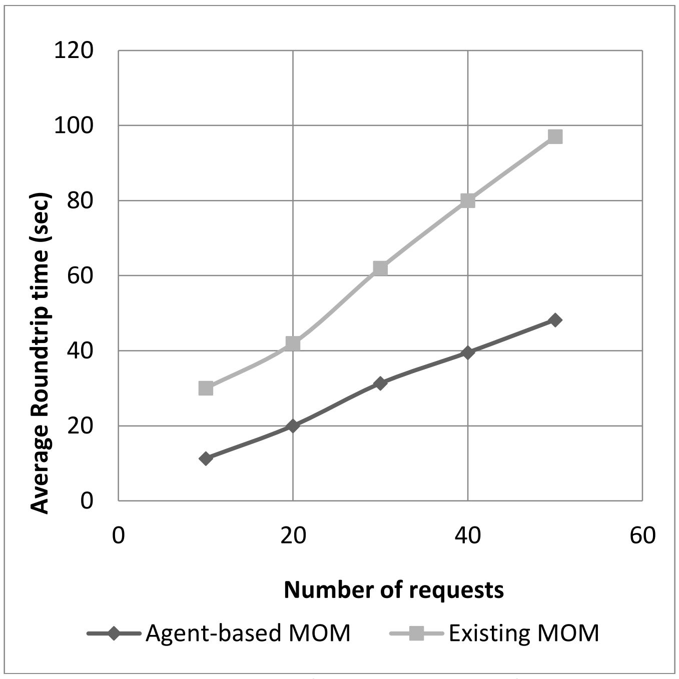


Fig. 7. Average round-trip time for agent-based MOM framework and Existing MOM (low level)

In Fig. 7 and Fig. 8, the changing of the average round-trip time at this specific value of the graph clearly shown. In the low level and the high level of traffic allocated, the figure of the average round-trip time changes due to the difference in the number of requests is different between the proposed agent-based MOM framework and the existing MOM. The proposed agent-based MOM framework was able to manage the request and provide the response by using lesser time than the existing MOM during the low level and the high level of traffic allocated.

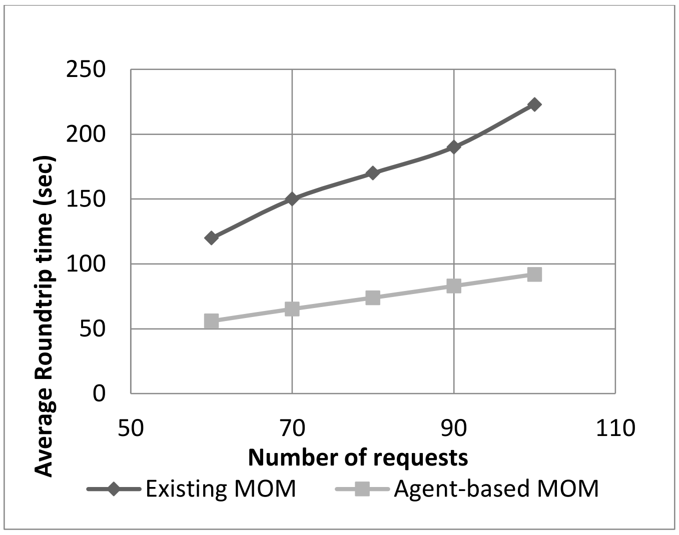


Fig. 8. Average round-trip time for Agent-based MOM Framework and Existing MOM (high level)

The linear pattern of the graph also had a substantial effect on the value of the average round-trip time and the number of requests allocated. As shown in Figure 7 and Figure 8, the number of requests assigned at the low level and the high level of request allocated were similar. The proposed agent-based MOM framework was able to achieve a more moderate and better average round-trip performance than the existing MOM. The difference in the average round-trip performance between the proposed agent-based MOM framework and the existing MOM in the low level and the high level of traffic was caused by the associated software components that were used to implement each application, which required different computation power and resources to execute the request and the respond in each communications scenario.

V. Conclusions

This paper presents the prototype implementation and the experimental simulation setup. The prototype implementation presents an overview of the tools and technologies used for prototyping. The multi-agent system was developed by using NetBeans and JADE platforms. The experimental setup was described in details for the experiment in Section 2. Subsequently, the simulation setup was described in details for implementation of the proposed work. Finally, the specific simulation scenario to be used for performance evaluation is presented.

This paper explored the details of the experiment results and the analysis of the results obtained for the proposed agent-based MOM framework. Then, the criteria that were used to evaluate the system were identified and were classified into two comparisons. Firstly, the performance analysis was conducted to assess the proposed agent-based MOM framework performance and with the existing MOM. Secondly, a comparative study between the proposed agent-based MOM framework and the JMS to evaluate and compare the multiplicity of cross-platform communications. The comparative research also conducted between the cross-platform communications and the multiplicity cross-platform communication of the proposed agent-based MOM framework.

The implementation results show that the proposed agent-based MOM framework was successful communicate between multiple types of SOA application with a better performance of the average of round-trip time where the proposed framework was successful in responding to all the requests. Additionally, the availability and communications scalability on the performance of the average round-trip time was intangible as well, which makes the proposed agent-based MOM framework an excellent solution to adopt for the cross-platform communications in SOA environment.

ACKNOWLEDGEMENT

We would like to express our gratitude to those that participate in this research

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest.

References

[1] M. Nikraz, G. Caire, and P.A. Bahri,. MAES: A Multi-Agent Systems Framework for Embedded Systems. Master's Thesis in Embedded Systems, Mekelweg 4, 2628 CD Delft, The Delft University of Technology Netherlands. (2017)

[2] T. Bayer, and C. Reich,. Security of Mobile Agents in Distributed Java Agent Development Framework (JADE) Platforms. ICONS 2017: The Twelfth International Conference on Systems. (2017)

[3] J. Board, (2008). JADE web services integration gateway (WSIG) guide , JADE WSIG Add-On GUIDE.

[4] X.T. Nguyen, , and R. Kowalczyk,. WS2JADE: Integrating Web Service with Jade Agents, in Verlag Berlin Heidelberg 2007, Springer: Berlin. (2007)

[5] W. Laftah, Z. Al-Yaseen, Ali Othman , and M. Z. Ahmad Nazri. A Large Data Exchange Method for Multi-agent in Java Agent Development Framework. Special Issue for "International Conference on Applied Science and Technology (ICAST), Malaysia. (2016)

[6] M. Deepa, and J. Punitha. A Hybrid Approach for Discovery of OWL-S Services Based on Functional and Non-Functional Properties, in WSEAS TRANSACTIONS on COMPUTERS, IEEE. (2015)

[7] N. Ibrahim M., I., and M.F Hassan, A Comprehensive Comparative Study of MOM for Adaptive Interoperability Communications in Service Oriented Architecture, International Journal of Trend in Scientific Research and Development (IJTSRD). (2019)

[8] *N. H, Alkahtania, S. Almohsen, N. M. Alkahtani, and G. A. Almalki. A Semantic Multi-Agent system to Exchange Information between Hospitals. The 8th International Conference on Ambient Systems, Networks and Technologies (ANT 2017).*

[9] N. Ibrahim M., I., and M.F Hassan, and M. H, Abdullah,. ABMOM for Cross-platform Communication in SOA Systems, 2013 International Conference on Research and Innovation in Information Systems (ICRIIS). Universiti Tenaga National, 2013, IEEE: KL. (2013)

[10] S. Pawar, and N. N. Chiplunkar. Discovery and Invocation of Web Services using Multi-Dimensional Data Model with WSDL, Indian Journal of Science and Technology, Vol 10(17), DOI: 10.17485/ijst/2017/v10i17/108890. (2017)

[11] M. Pfaff, and H. Krcmar, A web-based system architecture for ontology-based data integration in the domain of IT benchmarking. Enterprise Information Systems Volume 12, Issue 3. (2018).

[12] *S., Mishra, S., Malik, N.K., Jain, and S. Jain. A Realist Framework for Ontologies and the Semantic Web. Procedia Computer Science Volume 70, Pages 483-490. (2015).*

[13] M., Saturno, L. F. P., Ramos, Polatoa, F., Deschamps, F.,E. and Loures F. R. Evaluation of interoperability between automation systems using multi-criteria methods, 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, Modena, Italy. (2017).

[14] H., Bensag, M. Youssfi, ,and O. Bouattane, Efficient Model for Distributed Computing based on Smart Embedded Agent. (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 8, No. 2. (2017).

[15] N. M. Hamka, and R. Mohamad. OntoUji: Ontology to Evaluate Domain Ontology for Semantic Web Services Description, Jurnal Teknologi, Vol. 69, No. 6, pp. 21–26. (2014).